Agriculture Canada

Research Branch
Direction générale de la recherche

Technical Bulletin 1985-5E

APHID:

A data base program for aphid collections

Canada

86-01615/

The map on the cover has dots representing
Agriculture Canada research establishments.

APHID:

A data base program for aphid collections

YVAN PELLETIER
Research Station
Fredericton, New Brunswick

Research Branch
Agriculture Canada
1985

Copies of this publication are available from:
Yvan Pelletier
Research Station
Research Branch, Agriculture Canada
P.O. Box 20280
Fredericton, New Brunswick
E3B 4Z7

Produced by Research Program Service

© Minister of Supply and Services Canada 1985
Cat. No. A54-8/1985-5E
ISBN 0-662-14360-4

TABLE OF CONTENTS

INTRODUCTION 1

APHID:THE SYSTEM 2

APHID:THE DATA FILES 2

The aphid name file 2

The host-plant name file 5

The collection data file 5

The slide data file 8

APHID:OPERATION 8

Updating or correcting a file 8

Printing a file 14

Printing the labels 14

REFERENCE 15

ACKNOWLEDGMENTS 15

APPENDIX A.- LISTING OF PROGRAM 16

ABSTRACT

APHID is a computerized data base program for the management of an aphid collection data. The information is stored in four files representing the aphid names, the host-plant names, the field collection data and the identification data. A total of 30 variables make up the data base. Each file can be updated, corrected and printed. A print out of the labels for the slides can also be produced. Because APHID is written in Pascal it can be adapted to different microcomputer systems. A listing of the program is included.

RESUME

APHID est un système informatique de gérance de données d'une collection de pucerons. L'information est répartie en quatres filières informatiques représentant le nom des pucerons, le nom de la plante hôte, les données prises lors de la collecte et les données relatives à l'identification. Au total, 30 variables forment la banque de données. Chacune des filières informatiques peut être mise a jour, corrigée et imprimée. On peut aussi produire les étiquettes des lames. APHID est écrit en langage Pascal, ce qui le rend compatible avec la pluspart des micro-ordinateurs. Le listage du programme est inclus.

INTRODUCTION

The collection of aphid specimens requires the storage and the classification of information along with the specimens themselves. These data are important for taxonomic and ecological studies. The usefulness of the data depends on the ease of retreaval and sorting. Card indexes require the duplication of information common to many entries. With large numbers of specimens the effort and the time needed to maintain accurate information increase significantly. Changes in the taxonomic identity of any aphid species represent additional time requirements.

Computerized files reduce the time needed to create and update the information concerning an aphid collection. Raworth and Frazer (1976) described a computer program that produces indexes. Although useful, this system still requires that the usual aphid collection data base be maintained in parallel.

APHID is a program specifically designed for the management of an aphid collection data. Because the data have to be entered only once, the time required to create and maintain the files is reduced sharply . When the name of an aphid is changed only one step is needed to correct the whole data base . All the information necessary to an aphid collection can be stored and maintained using APHID. With its computerized data files, APHID allows you to use as many indexes as you wish.

You do not have to be a programmer to use APHID. The system is menu driven and easy to learn. It is fast too. Even though APHID is made up of several rather large Pascal programs, the slowest element is still the typist. A complete program listing is also provided so you can adapt APHID to your particular needs. The portability of the Pascal language makes APHID adaptable to different micro-computer or operating systems.

APHID has been developped on an Apple IIe equipped with an Apple 80-column video card and two floppy disk drives. Apple Pascal (128K) was used as operating system and language package. The printer was an Apple Imagewriter linked to the computer with an Apple Super Serial card. APHID can also run on a 64K system but more memory space is needed to edit the programs before compiling. The usage of other equipment may

require simple changes to APHID.

For simplicity, the following instructions presume that you are familiar with a Pascal operating system and that your computer system is the same than the one described above.

APHID:THE SYSTEM.

The Pascal programs making up APHID are stored in six files. Some of these programs use procedures from the Apple Pascal library. The screen control information is provided by the file SYSTEM.MISCINFO. The SYSTEM.FILEINFO file keeps APHID up to date on the names of the last APHID data files used.

The information is stored in four different APHID data files: aphid names, host-plant names, field collection data and identification data for each slide. This information can be easily updated or corrected. When new records are added to the aphid name or the host-plant name files, APHID will check if these records are not already on file. Only new names can be stored. APHID will also let you define synonyms.

APHID prints all or part of the information in its files. A listing of the aphid and host-plant names can be printed in alphabetical order with the synonyms or only the synonyms can be printed. APHID can also print the information in a 'label format' ready to be photo reduced and attached to your slides. This avoids transcription errors.

APHID:THE DATA FILES

APHID stores the information in four random access data files. You can give them the name of your choice. APHID will prompt you for the name of each file and he will refer to them using a standard record name. Names and descriptions of the files are listed in Table 1.

The aphid name file.

The record structure of the aphid name file is described in Table 2.

The information of the aphid name file is arranged in a binary tree with a synonym branch. Each record can be represented as a node in the tree (Fig 1). Each node is the intersection of four branches: one pointing to a lower node within the tree (root branch)(unless it is the base), one

Table 1.- the APHID's data files and record names.

PROMPT AS	RECORD	DESCRIPTION
Aphid name file	APHID	Aphid names and binary tree data
Host-plant file	PLANT	Host-plant names and binary tree data
Collection data file	COLL	Field collection data
Slide data file	SLIDE	Slide data

Table 2.- Record structure of the aphid name file.

NAMES	FORMAT	DESCRIPTION	INDEX **
ANUM	INTEGER[6]*	Reference number for the aphid name	3
AGENUS	STRING[20]	Aphid genus	4
ASBGN	STRING[20]	Aphid subgenus	5
ASPEC	STRING[20]	Aphid species	6
ASBSP	STRING[20]	Aphid subspecies	7
ADESC	STRING[20]	Aphid describer	8
ASIM	BOOLEAN	True if the aphid name is a synonym	SIM
ASYN	INTEGER	Reference number of a synonym	SYN
AHNOD	INTEGER	Branch to an aphid name with an higher alphabetical value	HN
ALNOD	INTEGER	Branch to an aphid name with a lower alphabetical value	LN
ABNOD	INTEGER	Branch to the aphid name placed one node lower	BN

* Number of digits of the variable.

** Refers to the index value of the variables DUM, DUML, LINES and OKSET and to the names of the temporary variables corresponding to the record variable in the COLLECT:SLIDE file

Fig 1.- Example of a binary tree arrangement.

```
          E     K
           \   /
            \ /
A--J--H      T
         \   /
          \ /
           N   (Base)
```

Each letter represents a node. The left side lines represent the low branch and the right side lines the high branch. Letters linked by horizontal lines represent a chain of synonyms.

pointing to a lateral node representing a synonym (synonym branch), and two pointing to nodes higher within the tree and corresponding to aphid names with lower (low branch) and higher (high branch) alphabetical values. Only the aphid genus name, species name and subspecies name are used to determine the position within the tree.

The program follows the tree from its base accordingly to the alphabetical value of the aphid name considered. If it is a new name it will be added at the top of the tree. For example if you want to add the name 'C' to the tree of the Fig 1, it will be compared with 'N', 'H' and 'E' successively. Having a lower value than 'E', it will be placed at the end of the low branch of 'E'. The program does not consider the synonyms present. If 'A' is entered as a new name, the program will add the record at the end of the low branch of 'E' even if 'A' is already in the tree as a synonym. It is the responsability of the user to keep track of the synonyms he creates. When you correct a name, the position of the record within the tree will also be corrected. For example, if you change the name 'T' for 'D', the program will change the position of the record to the low branch of 'E'.

Because of the structure of the file, the entire list of names must be stored in a single aphid name file. The first reference number for an aphid name must be 1 and it may not exceed 32766. One diskette (140 Kbytes) can handle 1149 aphid names.

The host-plant name file.

The host-plant name file is structured like the aphid name file. Only the names of the record variables are different (Table 3). All the limitations and specifications of the aphid name file apply to the host-plant file. The position of the record in the binary tree is determined accordingly to the alphabetical value of the genus name, species name and the variety name.

The collection data file.

The record structure of the collection data file is presented in Table 4.

Only 215 collection records can be stored on a diskette and the first collection reference number of a collection data file can be any integer smaller than 32767. You can number the collection records form 1 to 32766 consecutively and store them in different collection data files (diskettes).

Table 3.- Record structure of the host-plant name file

NAMES	FORMAT	DESCRIPTION	INDEX **
PNUM	INTEGER	Host-plant reference number	7
PFAM	STRING[20]*	Host-plant family	8
PGENUS	STRING[20]	Host-plant genus	9
PSPEC	STRING[20]	Host-plant species	10
PVAR	STRING[20]	Host-plant variety or cultivar	11
PDESC	STRING[20]	Host-plant describer	12
PSIM	BOOLEAN	True if the host-plant name is a synonym	SIM
PSYN	INTEGER	Reference number of a synonym	SYN
PHNOD	INTEGER	Branch to a host-plant name with higher alphabetical value	HN
PLNOD	INTEGER	Branch to a host-plant name with a lower alphabetical value	LN
PBNOD	INTEGER	Branch to the host-plant name placed one node lower	BN

* Number of digits of the variable.

** Refers to the index value of the variables DUM, DUML, LINES and OKSET and to the names of the temporary variables corresponding to the record variable in the COLLECT:HOST file.

Table 4.- Record structure of the collection data file.

NAMES	FORMAT	DESCRIPTION	INDEX **
CNUM	INTEGER	Collection data reference number	1
DATE	STRING[12]*	Date of the collect	2
CITY	STRING[15]	City where the collect took place	3
STATE	STRING[15]	Province where the collect took place	4
COUNTRY	STRING[15]	Country where the collect took place	5
COLLECT	STRING[20]	Name of the collector	6
PNUM	INTEGER	Reference number of the host-plant name	7
PCOMF	STRING[1]	Positive identification of the host-plant	13
LOCAT	STRING[50]	Location of the aphid on the host-plant	14
COLONY	STRING[50]	Size or characteristics of the aphid colony	15
COLOUR	STRING[226]	Colour of the aphid alive	16
CCOM	STRING[226]	Comments on the collect	17

* The number of digits of the record variable.

** Refers to the index value of the variables DUM, DUML, LINES and OKSET corresponding to the names of the record variables in the COLLECT:HOST file .

The slide data file.

The record structure of the slide data file is presented in Table 5.

Up to 1188 slide records can be stored on a diskette. The first slide reference number of a slide data file can be any integer smaller than 1,000,000. You can number the slide records from 1 to 999,999 and store them in different slide data files.

APHID: OPERATION.

The best way to begin a session is to boot the system with an APHID boot diskette . To create a boot diskette, the following files must be transferred to a blank diskette:SYSTEM.APPLE, SYSTEM.PASCAL, SYSTEM.MISCINFO, SYSTEM.LIBRARY, SYSTEM.SARTUP, FILEINFO, TITLE.CODE, SLIDE.CODE, HOST.CODE, HARDCOPY.CODE and LABELS.CODE. The volume name of the disk must be COLLECT:. With the boot diskette, APHID will be executed automatically. First, a welcome message will be printed on the screen. Press the space bar and the APHID's main menu will appear (Table 6). Each time you finish a subprogram, the main menu will be printed on the screen unless an input/output error occurred. To finish the session type 'Q'. You can answer to all APHID prompt line with lower or upper case characters. This also includes the input of file names. When prompted, it is sufficient to type the letter corresponding to your selection.

Updating or correcting a file.

If you choose to update or correct a file, the system will load the program and execute an initialization. Each time APHID uses a data file, it stores the name in COLLECT:FILEINFO. The initialization program will ask if you want to work on the file previously used. If you do so, just press 'y'. If you want to work on a different file press 'n' and enter the new file name. APHID will search for a file of the same name. If the corresponding volume is not on-line, the system will wait until you provide it and will then resume its search. If for some reason you cannot provide it, press CONTROL-RESET to reboot the system. If there is no file with the same name, APHID will ask if you want to change the file name, open a new file with this file name or return to the main menu. This procedure will be repeated for each file type needed by the

Table 5.- Record structure of the slide data file.

NAMES	FORMAT	DESCRIPTION	INDEX **
SNUM	INTEGER[6]*	Reference number of the slide data	1
CNUM	INTEGER	Reference number of the collection data	2
ANUM	INTEGER	Reference number of the aphid name	3
ACONF	STRING[1]	Positive identification of the aphid	9
MORPH	STRING[12]	Morph of the aphid	10
IDENT	STRING[20]	Name of the identifier	11
MANY	STRING[2]	Number of specimens on the slide	12
SCOM	STRING[65]	Comments on the slide	13

* Number of digits of the variable.

** Refers to the index value of the variables DUM, DUML, LINES and OKSET corresponding to the record variable in the COLLECT:SLIDE file.

Table 6.- Content of APHID's main menu.

A. Update the collection data file
B. Correct the collection data file
C. Update the slide data file
D. Correct the slide data file
E. Update the aphid name file
F. Correct the aphid name file
G. Update the host-plant name file
H. Correct the host-plant name file
I. Print a file
J. Print labels

program.

After the phase of initialization is completed, APHID will display a column in inverse video corresponding to the record variables of the file you are working with. If you are correcting a file (or updating the first record of a new collection data file or slide data file) the cursor will be positioned in the first line waiting for the reference number of the record to correct. The cursor will return to the first line if you enter a number that is out of range. It is obvious that you cannot correct a record of a new (empty) file. The program will try to access a record greater than 0 and such record does not exist. In that case, press CONTROL-RESET to reboot the system. When you update a file, APHID will provide the reference number automatically (except for the first reference number of a new file).

When correcting a file, APHID will display the present content of the record you requested . What you see on the screen as you make corrections, is what will be stored. During this process the APHID editor is in charge. The editor allows the correction or input of any variable at any time. You can move the cursor up one line, down one line, one character right or left by using the four arrow keys. RETURN will cause the cursor to move to the beginning of the next variable (not necessarily the next line) and cause an exit from the editor if the cursor is located within the last variable.

You can change the value of a variable by typing the new value (or characters) over the old one. One or many characters can be corrected at any position of any variable by moving the cursor over the character(s) to be changed and typing the correct one(s). You can delete a character by typing '/'. The character on wich the cursor is placed will be erased. If the cursor is past the last character, the last character will be deleted. You can erase the entire variable by moving the cursor to the first character of the variable and pressing SPACEBAR and RETURN. You can also insert one or more characters by typing '^' (SHIFT-6) and the caracter(s) to be inserted. You have then activated a sub-editor. It uses the same control characters than the editor (except up-arrow, down-arrow and insert) to correct the string of characters to be inserted. To exit the insertion mode type RETURN.

The number of digits and the set of characters that can be used is defined for each variable (Table 7). The APHID editor will not allow you to enter a string longer than the predefined maximum length. Each character entered must also be an acceptable one. For example, you cannot enter the letter 'a' when you are editing an aphid reference number because it is not in the set of acceptable characters (Table 7).

Table 7.- Set of usable characters (OKSET) of the editable record variables.

RECORD	VARIABLE	CHARACTERS
COLL	CNUM	NB
	DATE	NB I V X -
	CITY	UC LC SP , - .
	STATE	UC LC SP , - .
	COUNTRY	UC LC SP , - .
	COLLECT	UC LC SP , - .
	PNUM	NB
	PCOMF	?
	LOCAT	NB UC LC SP ! " # % & ' ( ) * + , - . : ; < = > ? @
	COLONY	NB UC LC SP ! " # % & ' ( ) * + , - . : ; < = > ? @
	COLOUR	NB UC LC SP ! " # % & ' ( ) * + , - . : ; < = > ? @
	CCOM	NB UC LC SP ! " # % & ' ( ) * + , - . : ; < = > ? @
PLANT	PNUM	NB
	PFAM	UC LC SP
	PGENUS	UC LC SP
	PSPEC	LC SP
	PVAR	UC LC SP
	PDESC	UC LC SP ( ) .
SLIDE	SNUM	NB
	CNUM	NB
	ANUM	NB
	ACOMF	?
	MORPH	UC LC SP , .
	IDENT	UC LC SP , . ( )
	MANY	NB
	SCOM	NB UC LC SP ! " # % & ' ( ) * + , - . : ; < = > ? @
APHID	ANUM	NB
	AGENUS	UC LC SP
	ASBGN	UC LC SP
	ASPEC	LC SP -
	ASBSP	UC LC SP
	ADESC	UC LC SP ( ) .

UC= upper case letters, LC= lower case letters,
NB= numbers, SP= space.

When updating or correcting a collection data file or a slide data file APHID will update the host-plant name file or the aphid name file at the same time. The aphid name or the host-plant name can be entered in different ways. You can enter only the reference number of the name and APHID will display the record on the screen after you have left the editor. First, APHID uses the reference number of the name to find the information on the name file. If the reference number of the name is out of range the system will use the last reference number used. If you do not remember the reference number or if it is a new name, delete the reference number and enter the name. APHID will search for the name in the name file and display the reference number.

When the record is entered, move the cursor to the last variable and press RETURN. You will leave the APHID editor and the information displayed on the screen will be processed. The program will verify if the aphid name or the host-plant name is new and has to be added to the list. Aphid names and host-plant names can be corrected only when this option is selected from the main menu. Otherwise, they can only be updated. Then, APHID will ask if you want to store the record. Answer by typing 'Y' or 'N'. The system will then ask if you want to process another record. Answer the prompt line as above.

It is important to realise that WHAT YOU SEE ON THE SCREEN IS WHAT WILL BE STORED. If you are updating a file the information you just typed will stay on the screen and will serve as basis for the next record. With the editor commands, you can correct only the variables that have to be changed.

When correcting an aphid name file or a host-plant name file, you can create a synonym by changing the previous name to the new one. APHID will search the file for the new name. You will then have three choices: 1) Create a synonym: the record of the old name is moved from its position in the binary tree to the synonym branch of the new name, 2) Erase the record:all variables are erased and the record of the old name removed from the binary tree, 3) Keep it unchanged:the record of the old name is left unchanged. When a synonym is defined, nothing of the old name record is changed but the variables defining its position in the binary tree. However, APHID will recognize it to be a synonym and will use the new name when printing the slide labels. The old and the new names will be printed on the listing of a file. If you correct an empty record (one that has been erased), only a new name will be accepted.

Printing a file.

When you choose to print a file from the main menu, the system will load the program and ask you wich file you want to print. The program will then list a series of options depending on the file requested. With the aphid name file and the host-plant name file, you can print the names in alphabetical order and by reference number or produce a list of the synonyms. You can print a whole file or only part of it, by reference number.

The files are linked by a reference number. The collection data record contains a variable representing the host-plant name (Table 4). The slide data record contains a variable representing the aphid name and one representing the collection data (Table 5). APHID can print only those reference numbers or the entire record they represent. The four files, sorted by slide reference number, can then be printed. However, to do so, the four files have to be on line.

APHID uses elite characters and produces a page with 60 lines (72 characters width) and an 1.5 in. left margin. It uses 8.5 x 11 in. paper.

Printing the labels.

APHID can print the information in a 'label' format, ready to be photo reduced and attached to the microscopic slides. For each desired slide number, the system will print two labels having nine lines of twenty characters maximum. One of the labels (left side) will include: the collection reference number, the location (country, province and city), the date, the name of the collector and the host-plant name (genus, species and variety). The other label (right side) will list: the slide reference number, the aphid name (genus, subgenus, species ,subspecies and the name of the describer) and the name of the identifier.

REFERENCE

Raworth D.A., Frazer B.D.. 1976. Compilation of taxonomic catalogues by computer. J. Entomol. Soc. Brit. Columbia. 73:63-67.

ACKNOWLEDGMENTS

I thank Dr. G. Boiteau who reviewed this manuscript. 'Apple', 'Apple Pascal', Apple Imagewriter' and 'Apple Super Serial' are trademarks of Apple Computer Inc.

APPENDIX A
THE LISTING OF THE APHID PROGRAMS

The next pages contain a annotated listing of the different Pascal programs making up APHID. A copy of the code files and/or text files is available from the author on request. The request must be accompaied by Apple compatible diskettes .

```
PROGRAM TITLE;
(*********************************************)
(*                                           *)
(* DISPLAY THE MAIN MENU AND GET THE SELECTION *)
(*                                           *)
(*********************************************)
USES CHAINSTUFF;
TYPE SETOFCHAR=SET OF CHAR;
VAR SEL:STRING;
    TEM:CHAR;


FUNCTION GETCHAR(OK:SETOFCHAR):CHAR;
(*************************************************)
(*                                               *)
(*      GET A CHARACTER FROM THE KEYBOARD.       *)
(*                                               *)
(*************************************************)
(* The function will accept the character if it is *)
(* present in the specified set of character (OK). *)
(* Otherwise, the function will produce a 'beep'   *)
(* and wait for another character.                 *)
VAR CH:CHAR;
  GOOD:BOOLEAN;
BEGIN
  REPEAT
    READ(KEYBOARD,CH);
    IF EOLN(KEYBOARD) THEN CH:=CHR(13);
    GOOD:=CH IN OK;
    IF NOT GOOD THEN WRITE(CHR(7))
      ELSE IF CH IN [' '..'#','%'..'.','0'..'Z','a'..'z'] THEN WRITE(CH);
  UNTIL GOOD;
  GETCHAR:=CH;
END;  (* GETCHAR *)


PROCEDURE SELECT;
(*********************************************)
(*                                           *)
(* PRINT THE MAIN MENU AND GET THE SELECTION *)
(*                                           *)
(*********************************************)
VAR I:INTEGER;
BEGIN
  WRITE(CHR(12));
  GOTOXY(35,0);
  WRITE(CHR(15),'MAIN MENU',CHR(14));
  GOTOXY(20,1);
  WRITE('*****************************************');
  FOR I:=1 TO 23 DO BEGIN
    GOTOXY(60,I);
    WRITE('*');
```

```
  END;
  GOTOXY(20,23);
  WRITE('****************************************');
  FOR I:=23 DOWNTO 1 DO BEGIN
    GOTOXY(20,I);
    WRITE('*');
  END;
  GOTOXY(22,2);
  WRITE('A. UPDATE THE COLLECTION DATA FILE.');
  GOTOXY(22,4);
  WRITE('B. CORRECT THE COLLECTION DATA FILE.');
  GOTOXY(22,6);
  WRITE('C. UPDATE THE SLIDE DATA FILE.');
  GOTOXY(22,8);
  WRITE('D. CORRECT THE SLIDE DATA FILE.');
  GOTOXY(22,10);
  WRITE('E. UPDATE THE APHID NAME FILE.');
  GOTOXY(22,12);
  WRITE('F. CORRECT THE APHID NAME FILE.');
  GOTOXY(22,14);
  WRITE('G. UPDATE THE HOST-PLANT NAME FILE.');
  GOTOXY(22,16);
  WRITE('H. CORRECT THE HOST-PLANT NAME FILE.');
  GOTOXY(22,18);
  WRITE('I. PRINT A FILE.');
  GOTOXY(22,20);
  WRITE('J. PRINT THE LABELS.');
  GOTOXY(22,22);
  WRITE('Enter your selection (Q)uit APHID): ');
  TEM:=GETCHAR(['A'..'J','a'..'j','q','Q']);
END;   (* SELECT *)


BEGIN   (* TITLE *)
  SELECT;
  (* Select the file containing the program to be executed. *)
  CASE TEM OF
    'A','B','G','H','a','b','g','h':SETCHAIN('COLLECT:HOST');
    'C','D','E','F','c','d','e','f':SETCHAIN('COLLECT:SLIDE');
    'I','i':SETCHAIN('COLLECT:HARDCOPY');
    'J','j':SETCHAIN('COLLECT:LABELS');
    'Q','q':SETCHAIN('');
  END;
  SEL:=' ';
  SEL[1]:=TEM;  (* Save the selection code. *)
  SETCVAL(SEL)
END.   (* TITLE *)
```

```
PROGRAM HOST;
(*********************************************)
(*                                           *)
(* UPDATE OR CORRECT THE COLLECTION DATA FILE *)
(* OR THE HOST-PLANT NAME FILE               *)
(*                                           *)
(*********************************************)
USES CHAINSTUFF;
TYPE SETOFCHAR=SET OF CHAR;
    CRTCOMMAND= (ERASEOS,ERASEOL,UP,DOWN,RIGHT,LEFT,LEADIN);
VAR COLL:FILE OF RECORD
           CNUM:INTEGER;
           DATE:STRING[12];
           CITY:STRING[15];
           STATE:STRING[15];
           COUNTRY: STRING[15];   (* See Table 4 for a description *)
           COLLECT:STRING[20];    (* of the variables.             *)
           PNUM:INTEGER;
           PCOMF:STRING[1];
           LOCAT:STRING[50];
           COLONY:STRING[50];
           COLOR:STRING[226];
           CCOM:STRING[226];
         END;
  PLANT:FILE OF PACKED RECORD
         PNUM:INTEGER;
         PFAM:STRING[20];
         PGENUS:STRING[20];
         PSPEC:STRING[20];      (* See Table 3 for a description *)
         PVAR:STRING[20];       (* of the variables.             *)
         PDESC:STRING[20];
         PSIM:BOOLEAN;
         PSYN:INTEGER;
         PHNOD:INTEGER;
         PLNOD:INTEGER;
         PBNOD:INTEGER;
       END;
   FNAMES:FILE OF RECORD        (* Structure of the FILEINFO file.*)
           CFILE,PFILE,         (* The record 0 is used to store  *)
           SFILE,AFILE:STRING;  (* the last file names used.      *)
         END;           (* CFILE=name of the collection data file *)
                        (* PFILE=name of the host-plant name file *)
                        (* SFILE=name of the slide data file      *)
                        (* AFILE=name of the aphid name file      *)

   DUM:ARRAY[1..17] OF STRING[226]; (* Working variable representing   *)
                                    (* the COLL and PLANT records.     *)
                                    (* Tables 3 and 4 list the index   *)
                                    (* values corresponding to the     *)
                                    (* records' variables.             *)

   LINES:ARRAY[1..17] OF INTEGER;   (* Vertical position on the screen *)
```

```
                                    (* of the first line of the COLL   *)
                                    (* and PLANT records' variables.   *)
                                    (* See Tables 3 and 4 for a        *)
                                    (* description of the index values.*)

   DUML:ARRAY[1..17] OF INTEGER;    (* Length of the COLL and PLANT    *)
                                    (* records' variables.  See Tables*)
                                    (* 3 and 4 for a description of    *)
                                    (* the index values.               *)

   OKSET:ARRAY[1..17] OF SETOFCHAR; (* Set of acceptable characters    *)
                                    (* of the COLL and PLANT records' *)
                                    (* variables (Table 7).  See       *)
                                    (* Tables 3 and 4 for a            *)
                                    (* description of the index values*)

   NEWREC, (* True if the host-plant name is not in file. *)
   SIM:BOOLEAN; (* True if the host-plant name is a synonym. *)
   LASTCOL, (* Number of records in the collection data file. *)
   CCONST,  (* Reference number (minus one) of the first record of *)
            (* the collection data file.                           *)
   LASTPL,  (* Number of records in the host-plant name file. *)
   I,       (* Counter. *)
   IORES:INTEGER;  (* I/O error code. *)
   LN,HN,BN,SYN,  (* Working variables of the PLANT variables defining *)
                  (* the position of the record within the binary tree.*)
   BASE:INTEGER;  (* Record number of the base of the binary tree. *)
   SEL,  (* Main menu selection code. *)
   BRANCH,  (* Define the branch where a new node will be added. *)
   WW:CHAR;  (* Dummy. *)
   CRTINFO: PACKED ARRAY[CRTCOMMAND] OF CHAR;  (* Screen commands. *)
   PREFIXED:ARRAY[CRTCOMMAND] OF BOOLEAN;
(*$I-*)


PROCEDURE GETCRTINFO;
(*****************************************)
(*                                       *)
(* READ THE SCREEN CONTROL CHARACTERS FROM *)
(* THE SYSTEM.MISCINFO FILE              *)
(*                                       *)
(*****************************************)
VAR BUFFER: PACKED ARRAY[0..511] OF CHAR;
    I,BYTE: INTEGER;
    F: FILE;
BEGIN
   RESET(F,'*SYSTEM.MISCINFO');
   I:=BLOCKREAD(F,BUFFER,1);
   CLOSE(F);
   BYTE:=ORD(BUFFER[72]);
   CRTINFO[LEADIN]:=BUFFER[62];  PREFIXED[LEADIN]:=FALSE;
   CRTINFO[ERASEOS]:=BUFFER[64]; PREFIXED[ERASEOS]:=ODD(BYTE DIV 8);
```

```
  CRTINFO[ERASEOL]:=BUFFER[65]; PREFIXED[ERASEOL]:=ODD(BYTE DIV 4);
  CRTINFO[RIGHT]:=BUFFER[66];   PREFIXED[RIGHT]:=ODD(BYTE DIV 2);
  CRTINFO[UP]:=BUFFER[67];      PREFIXED[UP]:=ODD(BYTE);
  CRTINFO[LEFT]:=BUFFER[68];    PREFIXED[LEFT]:=ODD(BYTE DIV 32);
  CRTINFO[DOWN]:=CHR(10);       PREFIXED[DOWN]:=FALSE;
END;  (* GETCRTINFO *)


PROCEDURE CRT(C:CRTCOMMAND);
(*************************)
(*                       *)
(*    SCREEN COMMANDS    *)
(*                       *)
(*************************)
BEGIN
IF PREFIXED[C] THEN UNITWRITE(1,CRTINFO[LEADIN],1,0,12);
  UNITWRITE(1,CRTINFO[C],1,0,12);
END;  (* CRT *)


PROCEDURE BYEBYE;
(****************************************)
(*                                      *)
(* UPDATE FILEINFO AND CLOSE THE FILES *)
(*                                      *)
(****************************************)
BEGIN
  SEEK(FNAMES,0);
  PUT(FNAMES);
  CLOSE(PLANT,LOCK);
  CLOSE(COLL,LOCK);
  CLOSE(FNAMES,LOCK);
  EXIT(PROGRAM);
END;  (* BYEBYE *)


FUNCTION TRANSF(Z:STRING):INTEGER;
(************************************************)
(*                                              *)
(* CONVERT THE VALUE OF A STRING INTO AN INTEGER *)
(*                                              *)
(************************************************)
VAR I:CHAR;
    COUNT,TEMP,J:INTEGER;
BEGIN
  IF (LENGTH(Z)=0) OR ((LENGTH(Z))=5) AND (Z)'32766')) THEN BEGIN
    TRANSF:=0;      (* Return 0 if the value of the string cannot  *)
    EXIT(TRANSF);   (* be converted into an integer.               *)
  END;
  TEMP:=0;
  FOR J:=1 TO LENGTH(Z) DO
```

```
    BEGIN
      COUNT:=0;
      IF Z[J]<>'0' THEN BEGIN
        COUNT:=ORD(Z[J])-48;
        TEMP:=TEMP+(COUNT*TRUNC(PWROFTEN(LENGTH(Z)-J)));
      END;
    END;
  TRANSF:=TEMP;
END; (* TRANSF *)


PROCEDURE DISP_B;
(*********************************************)
(*                                           *)
(* DISPLAY THE INVERSE VIDEO LEFT COLUMN FOR *)
(* THE COLLECTION DATA VARIABLES             *)
(*                                           *)
(*********************************************)
BEGIN
   WRITE(CHR(15));
   GOTOXY(0,LINES[1]);WRITE('COLLECTION #:');
   GOTOXY(0,LINES[2]);WRITE('DATE:.......:');
   GOTOXY(0,LINES[3]);WRITE('CITY:.......:');
   GOTOXY(0,LINES[4]);WRITE('PROVINCE:....');
   GOTOXY(0,LINES[5]);WRITE('COUNTRY:.....');
   GOTOXY(0,LINES[6]);WRITE('COLLECTOR:...');
   GOTOXY(0,LINES[13]);WRITE('POSITIVE ID?:');
   GOTOXY(0,LINES[14]);WRITE('LOCATION:....');
   GOTOXY(0,LINES[15]);WRITE('COLONY SIZE:.');
   GOTOXY(0,LINES[16]);WRITE('COLOR:.......');
   GOTOXY(0,LINES[17]);WRITE('COMMENTS:....');
   WRITE(CHR(14));
END;  (* DISP_B *)


PROCEDURE DISP_C;
(*********************************************)
(*                                           *)
(* DISPLAY THE INVERSE VIDEO LEFT COLUMN FOR *)
(* THE HOST-PLANT NAME VARIABLES             *)
(*                                           *)
(*********************************************)
BEGIN
  WRITE(CHR(15));
  GOTOXY(0,LINES[7]);WRITE('PLANT NUMBER:');
  GOTOXY(0,LINES[8]);WRITE('PLANT FAMILY:');
  GOTOXY(0,LINES[9]);WRITE('PLANT GENUS:.');
  GOTOXY(0,LINES[10]);WRITE('PLANT SP.:...');
  GOTOXY(0,LINES[11]);WRITE('PLANT VAR.:..');
  GOTOXY(0,LINES[12]);WRITE('PLANT DESCR.:');
  WRITE(CHR(14));
END;  (* DISP_C *)
```

```
FUNCTION GETCHAR(OK:SETOFCHAR):CHAR;
(*************************************)
(*                                   *)
(* GET A CHARACTER FROM THE KEYBOARD *)
(*                                   *)
(*************************************)

(* See the program 'TITLE' for the listing and comments. *)


PROCEDURE GETSTRING(S,E:INTEGER);
(*****************************)
(*                           *)
(*       APHID EDITOR        *)
(*                           *)
(*****************************)
VAR S1:STRING[1];      (* Character entered via GETCHAR. *)
    ST:STRING[227];    (* Working variable. *)
    I,                 (* Counter. *)
    LO,LOC:INTEGER;    (* Position of the cursor within the variable *)
                       (* (LOC) and the string to be inserted (LO).  *)


PROCEDURE INS;
(*********************************)
(*                               *)
(* INSERTION MODE AND SUB-EDITOR *)
(*                               *)
(*********************************)
VAR S2:STRING[1];      (* Character inputed via GETCHAR. *)
BEGIN
  IF (LENGTH(DUM[I])=DUML[I]) OR (LENGTH(DUM[I])<LOC) THEN WRITE(CHR(7))
                       (* Insertion is not possible if the variable is *)
                       (* full or the cursor is located after the last *)
                       (* character of the variable.                   *)
  ELSE BEGIN
    ST:='';LO:=1;S2:=' ';
    REPEAT
      (* Definition of the set of characters to be used by GETCHAR. *)
      IF LENGTH(ST)=0 THEN S2[1]:=GETCHAR(OKSET[I]+[CHR(13)])
        (* Delete mode cannot be used if the string to be   *)
        (* inserted is empty.                               *)
      ELSE IF (LENGTH(ST)+LENGTH(DUM[I]))=DUML[I] THEN
              S2[1]:=GETCHAR([CHR(13),'/'])
           (* No more character can be inserted if the variable    *)
           (* reached is maximum length.  Characters can be erased.*)
           ELSE S2[1]:=GETCHAR(OKSET[I]+[CHR(13),'/']);
      IF S2[1]='/'THEN BEGIN     (* Delete mode. *)
          CRT(LEFT);WRITE(' ');CRT(LEFT);
```

```
          DELETE(ST,LENGTH(ST),1);
          LO:=LO-1;
        END;
        IF S2[1] IN OKSET[I] THEN BEGIN
          (* Add the character to the working string and print the   *)
          (* new value of the variable.                              *)
          GOTOXY(13,LINES[I]);ST:=CONCAT(ST,S2);
          LO:=LO+1;
          WRITE((COPY(DUM[I],1,LOC-1)),ST,(COPY(DUM[I],LOC,
               (LENGTH(DUM[I])-(LOC-1)))));
          CRT(ERASEOL);
          (* Move the cursor to the proper location. *)
          IF (11+LOC+LO)>160 THEN GOTOXY(LOC+LO-149,LINES[I]+2)
          ELSE IF (11+LOC+LO)>80 THEN GOTOXY(LOC+LO-69,LINES[I]+1)
               ELSE GOTOXY(11+LOC+LO,LINES[I]);
        END;
      UNTIL S2[1]=CHR(13);  (* Insertion mode ends with a RETURN. *)
      (* The value of the variable is changed and displayed. *)
      IF (LENGTH(ST))>0 THEN INSERT(ST,DUM[I],LOC);
      GOTOXY(13,LINES[I]);
      WRITE(DUM[I]);LOC:=LOC+LO-1;CRT(ERASEOL);
      IF (12+LOC)>160 THEN GOTOXY(LOC-148,LINES[I]+2)
      ELSE IF (12+LOC)>80 THEN GOTOXY(LOC-68,LINES[I]+1)
           ELSE GOTOXY(12+LOC,LINES[I]);
    END;
  END;  (* INS *)


  PROCEDURE JUMUP;
  (************************************)
  (*                                  *)
  (* MOVE THE CURSOR TO THE LINE ABOVE *)
  (*                                  *)
  (************************************)
  (* The cursor will move to:                                        *)
  (*      the same horizontal position if the line above reaches     *)
  (*           its position,                                         *)
  (*      the last character if the line above does not reach        *)
  (*           the horizontal position of the cursor,                *)
  (*      the first position if the variable is empty.               *)
  (* Empty lines of a variable will be skipped.                      *)
  VAR C,A,LI:INTEGER;
  BEGIN
    IF LOC<68 THEN BEGIN   (* If the cursor is located on the first line  *)
                           (* of the variable.                           *)
      IF I=S THEN WRITE(CHR(7))  (* 'Beep' if trying to jump above the    *)
      ELSE BEGIN                 (*  first line of the first variable.    *)
        I:=I-1;
        A:=LENGTH(DUM[I]);
        IF A>147 THEN BEGIN            (* If the variable above occupies *)
          LI:=2;                       (* three lines.                   *)
          IF A<160+LOC THEN LOC:=A
```

```
        ELSE LOC:=160+LOC;
        C:=LOC-148;
      END
      ELSE IF A>67 THEN BEGIN          (* If the variable above occupies *)
             LI:=1;                    (* two lines.                     *)
             IF A<80+LOC THEN LOC:=A
             ELSE LOC:=80+LOC;
             C:=LOC-68;
           END
           ELSE BEGIN                  (* If the variable above occupies *)
             LI:=0;                    (* one line.                      *)
             IF A<LOC THEN LOC:=A;
             IF LOC<1 THEN LOC:=1;
             C:=LOC+12;
           END;
           GOTOXY(C,LINES[I]+LI);
        END;
      END
    ELSE BEGIN    (* If the cursor is not located on the first line. *)
      IF LOC>147 THEN BEGIN      (* If the cursor is on the third line *)
        LI:=1;                   (* it will move to the same location  *)
        LOC:=LOC-80;             (* on the second line.                *)
        C:=LOC-68;
      END
      ELSE BEGIN                 (* If the cursor is located on the    *)
        LI:=0;                   (* second line, it will move to the   *)
        IF LOC<81 THEN BEGIN     (* same location on the first line or *)
          LOC:=1;                (* to the first character of the      *)
          C:=13;                 (* variable if the cursor is under the*)
        END                      (* inverse video left column.         *)
        ELSE BEGIN
          LOC:=LOC-80;
          C:=LOC+12;
        END;
      END;
      GOTOXY(C,LINES[I]+LI);
    END;
  END;  (* JUMUP *)


PROCEDURE JUMDOWN;
(*************************************)
(*                                   *)
(* MOVE THE CURSOR TO THE LINE BELOW *)
(*                                   *)
(*************************************)
(* The cursor will move to:                                          *)
(*      the same horizontal position if the line below reaches       *)
(*           its position,                                           *)
(*      the last character if the line below does not reach the      *)
(*           position of the cursor,                                 *)
(*      the first position if the variable is empty.                 *)
```

```
(* Empty lines of a variable will be skipped.                          *)
VAR C,A,LI:INTEGER;
BEGIN
  A:=LENGTH(DUM[I]);
  IF (A<68) OR (LOC>147) OR ((67<LOC) AND (LOC<147) AND (A<148)) THEN BEGIN
    (* If the cursor is on the last line of the variable. *)
    IF I=E THEN BEGIN
      WRITE(CHR(7));  (* Can not jump below the last line of the *)
      EXIT(JUMDOWN);  (* last variable.                          *)
    END;
    I:=I+1;
    A:=LENGTH(DUM[I]);
    IF LOC>67 THEN BEGIN                (* If the cursor is not on the  *)
      IF LOC>147 THEN LOC:=LOC-160      (* first line of the variable.  *)
      ELSE IF LOC>67 THEN LOC:=LOC-80;
      IF LOC<13 THEN LOC:=1;
    END;
    IF LOC>A THEN LOC:=A;
    IF LOC<1 THEN LOC:=1;
    LI:=0;
    C:=LOC+12;
  END
  ELSE BEGIN  (* If the cursor is not on the last line of the variable.*)
    LOC:=LOC+80;
    IF LOC>A THEN LOC:=A;
    IF LOC>147 THEN BEGIN
      LI:=2;
      C:=LOC-148;
    END
    ELSE BEGIN
      LI:=1;
      C:=LOC-68;
    END;
  END;
  GOTOXY(C,LINES[I]+LI);
END;  (* JUMDOWN *)


BEGIN  (* GETSTRING *)
  FOR I:=S TO E DO BEGIN  (* Edit the strings indexed S to E of the *)
                          (* working variable.                      *)
    S1:=' ';LOC:=1;
    GOTOXY(13,LINES[I]);
    REPEAT
      IF LOC<=DUML[I] THEN S1[1]:=GETCHAR(OKSET[I]+[CHR(13),CHR(10),
                                  CHR(11),CHR(8),CHR(21),'/','^'])
      ELSE S1[1]:=GETCHAR([CHR(13),CHR(10),CHR(11),CHR(8),
                          CHR(21),'/','^']);
       (* Check if the variable reached its maximum length. *)
      IF S1[1]=CHR(8) THEN BEGIN     (* Move the cursor one character *)
        IF LOC=1 THEN WRITE(CHR(7))  (* to the left.  'Beep' if the   *)
        ELSE BEGIN                   (* cursor is at the first        *)
```

```
    WRITE(CHR(8));LOC:=LOC-1;  (* position of the variable.     *)
  END;
END;

IF S1[1]=CHR(21) THEN BEGIN    (* Move the cursor one character *)
  IF LOC>LENGTH(DUM[I]) THEN   (* to the right.  'Beep' if the  *)
    WRITE(CHR(7))              (* cursor is after the last      *)
  ELSE BEGIN                   (* character of the variable.    *)
    CRT(RIGHT);LOC:=LOC+1;
    IF (LOC=68) OR (LOC=148) THEN CRT(DOWN);
  END;
END;

IF S1[1]='/' THEN   (* Delete mode:                              *)
                    (* 'Beep' if the variable is empty.          *)
                    (* Erase the character on wich the cursor    *)
                    (* is located or the last character of the   *)
                    (* variable if the cursor is after the last  *)
                    (* character.                                *)
  IF LENGTH(DUM[I])=0 THEN WRITE(CHR(7))
  ELSE BEGIN
    IF (LOC-1)=LENGTH(DUM[I]) THEN BEGIN
      CRT(LEFT);WRITE(' ');CRT(LEFT);
      DELETE(DUM[I],LENGTH(DUM[I]),1);
      LOC:=LOC-1;
    END
    ELSE BEGIN
      DELETE(DUM[I],LOC,1);
      GOTOXY(13,LINES[I]);
      WRITE(DUM[I]);CRT(ERASEOL);
      IF (LOC>148) THEN GOTOXY(LOC-148,LINES[I]+2)
      ELSE IF (LOC>68) THEN GOTOXY(LOC-68,LINES[I]+1)
           ELSE GOTOXY(12+LOC,LINES[I]);
    END;
  END;

IF S1[1]='^' THEN INS;  (* Insert mode *)

IF S1[1]=CHR(10) THEN JUMDOWN;  (* Move the cursor on the line  *)
                                (* below.                        *)

IF S1[1]=CHR(11) THEN JUMUP;  (* Move the cursor on the line *)
                              (* above.                      *)

IF S1[1] IN OKSET[I] THEN BEGIN     (* Change the character on wich *)
  IF LOC<=LENGTH(DUM[I]) THEN BEGIN (* the cursor is located or add *)
    DUM[I][LOC]:=S1[1];             (* a character at the end of the*)
  END                               (* string.                      *)
  ELSE BEGIN
    DUM[I]:=CONCAT(DUM[I],S1);
  END;
  LOC:=LOC+1;
```

```
      END;

  UNTIL S1[1]=CHR(13);  (* Process the variable and jump to the next. *)
  IF DUM[I]<>'' THEN IF DUM[I][1]=' ' THEN DUM[I]:='';
  IF DUML[I]>67 THEN BEGIN       (* Erase the variable if the first  *)
    GOTOXY(0,LINES[I]+1);        (* character is a blanck.           *)
    CRT(ERASEOL);
  END;
  IF DUML[I]>147 THEN BEGIN
    GOTOXY(0,LINES[I]+2);
    CRT(ERASEOL);
  END;
  GOTOXY(13,LINES[I]);        (* Display the variable on the screen. *)
  WRITE(DUM[I]);
  CRT(ERASEOL);
  END;
END;  (* GETSTRING *)


FUNCTION SIMIL (A,B:STRING):INTEGER;
(*************************************************)
(*                                               *)
(* COMPUTE THE DIFFERENCE (NUMBER OF MATCHLESS   *)
(* CHARACTERS) BETWEEN TWO STRINGS               *)
(*                                               *)
(*************************************************)
VAR TEM,I,AL:INTEGER;
BEGIN
  IF LENGTH(A)>LENGTH(B) THEN BEGIN
    AL:=LENGTH(B);
    TEM:=LENGTH(A)-LENGTH(B);        (* Find the shortest string. *)
  END
  ELSE BEGIN
    AL:=LENGTH(A);
    TEM:=LENGTH(B)-LENGTH(A);
  END;
  IF AL>0 THEN FOR I:=1 TO AL DO     (* Compare two strings      *)
    IF A[I]<>B[I] THEN TEM:=TEM+1;   (* character by character.  *)
  SIMIL:=TEM;
END;  (* SIMIL *)


FUNCTION IOCHECK(F:STRING):BOOLEAN;
(*************************************)
(*                                   *)
(* VERIFY THE STATUS OF THE LAST I/O *)
(* OPERATION AND PRINT AN ERROR      *)
(* MESSAGE IF THE OPERATION WAS NOT  *)
(* SUCCESSFUL                        *)
(*                                   *)
(*************************************)
```

```
BEGIN
  IORES:=IORESULT;
  IF IORES=0 THEN IOCHECK:=TRUE
  ELSE BEGIN
    IOCHECK:=FALSE;
    CASE IORES OF
      2,3,7,10: BEGIN
                  GOTOXY(0,22);
                  WRITE(CHR(15),'ERROR IN FILENAME ',F,CHR(14));
                  CRT(ERASEOL);
                END;
      5,6,9:    BEGIN
                  GOTOXY(0,21);
                  WRITE(CHR(15),F,' NOT ON LINE.  CORRECT AND PRESS SPACEBAR.',
                        CHR(14));
                  CRT(ERASEOL);
                  WW:=GETCHAR([' ']);
                END;
      8,16:     BEGIN
                  GOTOXY(0,21);
                  WRITELN(CHR(15),'UNABLE TO WRITE DATA IN ',F,CHR(14));
                  CRT(ERASEOL);
                  WRITE(CHR(15),'Press spacebar to continue.',CHR(14));
                  WW:=GETCHAR([' ']);
                END;
      11:       BEGIN
                  GOTOXY(0,22);
                  WRITE(CHR(15),F,' ALREADY EXIST.  CHANGE FILE NAME.',CHR(14));
                  CRT(ERASEOL);
                END;
      12,13,14,15,64: BEGIN
                  GOTOXY(0,22);
                  WRITE(CHR(15),'INPUT/OUTPUT ERROR #',IORES,CHR(14));
                  CRT(ERASEOL);
                  SETCHAIN('');
                  BYEBYE;
                END;
    END;
  END;
END;  (* IOCHECK *)


PROCEDURE INIT;
(************************)
(*                      *)
(* INITIALIZATION STEPS *)
(*                      *)
(************************)
VAR I:INTEGER;
```

```
PROCEDURE INITCOL;
(*********************************************)
(*                                           *)
(* INITIALIZATION OF THE VARIABLES NEEDED TO *)
(* WORK ON THE COLLECTION DATA FILE          *)
(*                                           *)
(*********************************************)
BEGIN
  DUML[1]:=5;
  DUML[2]:=12;
  FOR I:=3 TO 5 DO DUML[I]:=15;
  DUML[6]:=20;
  DUML[13]:=1;
  FOR I:=14 TO 15 DO DUML[I]:=50;
  FOR I:=16 TO 17 DO DUML[I]:=226;
  OKSET[1]:=['0'..'9'];
  OKSET[2]:=OKSET[1]+['-','I','V','X'];
  FOR I:=3 TO 6 DO OKSET[I]:=['A'..'Z','a'..'z',',','-','.',' '];
  OKSET[13]:=['?'];
  FOR I:=14 TO 17 DO OKSET[I]:=[' '..'#','%'..'.','0'..'Z','a'..'z'];
END;  (* INITCOL *)


PROCEDURE INITPL;
(*********************************************)
(*                                           *)
(* INITIALIZATION OF THE VARIABLES NEEDED TO *)
(* WORK ON THE HOST-PLANT NAME FILE          *)
(*                                           *)
(*********************************************)
BEGIN
  DUML[7]:=5;
  FOR I:=8 TO 12 DO DUML[I]:=20;
  OKSET[7]:=['0'..'9'];
  OKSET[8]:=['A'..'Z','a'..'z',' '];
  OKSET[9]:=OKSET[8];
  OKSET[10]:=['a'..'z',' '];
  OKSET[11]:=OKSET[8];
  OKSET[12]:=OKSET[11]+['(',')','.'];
END;  (* INITPL *)


PROCEDURE OPEN (A:CHAR;L:INTEGER);
(*************************************)
(*                                   *)
(*       NAME AND OPEN FILES         *)
(*                                   *)
(*************************************)
(* The record 0 of each file contains information about the file.  *)
(* For the collection data file:                                   *)
(*      COLL^.CNUM=number of records in the file                   *)
```

```
(*      COLL^.PNUM=first collection reference number (minus one )  *)
(*                  in the file                                    *)
(*      COLL^.CITY=collection data file identification code ('C')  *)
(* For the host-plant name file:                                   *)
(*      PLANT^.PNUM=number of records in the file                  *)
(*      PLANT^.PBNOD=record number of the base of the binary tree  *)
(*      PLANT^.PFAM=host-plant name file identification code ('P') *)

VAR OLD,RES,CH:BOOLEAN;
    CHOI:CHAR;


PROCEDURE OP1;
(*********************************)
(*                               *)
(*      GET THE FULL FILE NAME   *)
(*                               *)
(*********************************)
(* Propose the last file name used.  If the name is not accepted, *)
(* read the new file name and update the proper FILEINFO record   *)
(* variable.                                                      *)
VAR RENAME:BOOLEAN;
BEGIN
  GOTOXY(0,L);CRT(ERASEOL);
  CASE A OF
    'A':BEGIN
          WRITE(CHR(15),'Collection data file=',CHR(14));
          IF (FNAMES^.CFILE='') OR (CH=TRUE) THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.CFILE,'  ');
            RENAME:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(22,L);CRT(ERASEOL);
            READLN(FNAMES^.CFILE);
          END;
        END;
    'B':BEGIN
          WRITE(CHR(15),'Host-plant name file=',CHR(14));
          IF (FNAMES^.PFILE='') OR (CH=TRUE) THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.PFILE,'  ');
            RENAME:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(22,L);CRT(ERASEOL);
            READLN(FNAMES^.PFILE);
          END;
        END;
  END;
END;  (* OP1 *)
```

```
BEGIN  (* OPEN *)
  OLD:=TRUE;
  CH:=FALSE;
  OP1;
  REPEAT
    IF CH=TRUE THEN OP1;
    IF OLD THEN BEGIN
      CASE A OF
        'A':RESET(COLL,FNAMES^.CFILE);
        'B':RESET(PLANT,FNAMES^.PFILE);
      END;
    END
    ELSE BEGIN
      CASE A OF
        'A':REWRITE(COLL,FNAMES^.CFILE);
        'B':REWRITE(PLANT,FNAMES^.PFILE);
      END;
    END;
    RES:=IOCHECK(' ');
    IF NOT (IORES IN [5,6,9]) THEN BEGIN
      IF RES=FALSE THEN BEGIN
        GOTOXY(0,L+2);CRT(ERASEOL);
        WRITE(CHR(15),'Choose between: C)hange the file name, N)ew file,',
              ' R)eturn to main menu.',CHR(14),' ');
        CHOI:=GETCHAR(['C','N','R','c','n','r']);
        CASE CHOI OF
          'C','c':CH:=TRUE;
          'N','n':BEGIN
                    CH:=FALSE;
                    OLD:=FALSE;
                  END;
          'R','r':BYEBYE;
        END;
      END
      ELSE BEGIN
        IF OLD THEN BEGIN  (* Verify if the file opened corresponds *)
          CASE A OF        (* to the desired type of file.          *)
            'A':BEGIN
                  IF COLL^.CITY<>'C' THEN BEGIN
                    GOTOXY(0,L+2);CRT(ERASEOL);
                    WRITE(CHR(15),FNAMES^.CFILE,' is not',
                          ' a collection data file.',CHR(14));
                    CH:=TRUE;RES:=FALSE;
                    CLOSE(COLL,LOCK);
                  END
                  ELSE BEGIN  (* Read the information in record 0. *)
                    LASTCOL:=COLL^.CNUM;
                    CCONST:=COLL^.PNUM;
                  END;
                END;
```

```
            'B':BEGIN
                  IF PLANT^.PFAM<>'P' THEN BEGIN
                    GOTOXY(0,L+2);CRT(ERASEOL);
                    WRITE(CHR(15),FNAMES^.PFILE,' is not',
                          ' a host-plant name file.',CHR(14));
                    CH:=TRUE;RES:=FALSE;
                    CLOSE(PLANT,LOCK);
                  END
                  ELSE BEGIN  (* Read the information in record 0. *)
                    LASTPL:=PLANT^.PNUM;
                    BASE:=PLANT^.PBNOD;
                  END;
                END;
          END;
        END
        ELSE BEGIN  (* If a new file. *)
          CASE A OF
            'A':BEGIN
                  LASTCOL:=0;
                  CCONST:=0;
                END;
            'B':BEGIN
                  LASTPL:=0;
                  BASE:=1;
                END;
          END;
        END;
      END;
    END;
  UNTIL RES;
END;  (* OPEN *)


BEGIN (* INIT *)
  WRITE(CHR(12));
  GOTOXY(30,0);
  WRITE(CHR(15),'INITIALIZATION',CHR(14));
  CASE SEL OF
    'A','B','a','b':BEGIN
                      FOR I:=1 TO 16 DO LINES[I]:=I;
                      LINES[17]:=19;
                      INITCOL;
                      INITPL;
                      OPEN('A',2);
                      OPEN('B',4);
                    END;
    'G','H','g','h':BEGIN
                      FOR I:=1 TO 6 DO LINES[I+6]:=I+5;
                      INITPL;
                      OPEN('B',2);
                    END;
```

```
  END;
  FOR I:=1 TO 17 DO DUM[I]:='';
  WRITE(CHR(12));
  LN:=0;BN:=0;HN:=0;SYN:=0;SIM:=FALSE;
END;


PROCEDURE STORECOL;
(*********************************************)
(*                                           *)
(* SAVE A RECORD INTO THE COLLECTION DATA FILE *)
(*                                           *)
(*********************************************)
BEGIN
  WITH COLL^ DO BEGIN       (* Transfer the values of the working *)
    CNUM:=TRANSF(DUM[1]);  (* variable to the record variables.  *)
    DATE:=DUM[2];
    CITY:=DUM[3];
    STATE:=DUM[4];
    COUNTRY:=DUM[5];
    COLLECT:=DUM[6];
    PNUM:=TRANSF(DUM[7]);
    PCOMF:=DUM[13];
    LOCAT:=DUM[14];
    COLONY:=DUM[15];
    COLOR:=DUM[16];
    CCOM:=DUM[17];
  END;
  IF LASTCOL=0 THEN CCONST:=COLL^.CNUM-1; (* Initialize CCONST if   *)
                                          (* working on a new file. *)
  SEEK(COLL,(COLL^.CNUM-CCONST));
  PUT(COLL);   (* Save the record. *)
  IF IOCHECK('COLLECTION FILE') THEN BEGIN (* Verify the status of   *)
                                           (* the I/O operation.     *)
    IF LASTCOL<(COLL^.CNUM-CCONST) THEN BEGIN
      SEEK(COLL,0);            (* Update the record 0, if a new   *)
      LASTCOL:=LASTCOL+1;      (* record was saved.               *)
      COLL^.CNUM:=LASTCOL;
      COLL^.PNUM:=CCONST;
      COLL^.CITY:='C';
      PUT(COLL);
    END;
  END;
END;  (* STORECOL *)
```

```
PROCEDURE STOREPL;
(*********************************************)
(*                                           *)
(* SAVE A RECORD INTO THE HOST-PLANT NAME FILE *)
(*                                           *)
(*********************************************)
BEGIN
  WITH PLANT^ DO BEGIN
    PNUM:=TRANSF(DUM[7]);  (* Transfer the values of the working *)
    PBNOD:=BN;             (* variable to the record variables.  *)
    PLNOD:=LN;
    PHNOD:=HN;
    PSYN:=SYN;
    PSIM:=SIM;
    PGENUS:=DUM[9];
    PFAM:=DUM[8];
    PSPEC:=DUM[10];
    PVAR:=DUM[11];
    PDESC:=DUM[12];
  END;
  SEEK(PLANT,PLANT^.PNUM);  (* Save the record. *)
  PUT(PLANT);
  IF IOCHECK('PLANT FILE')THEN BEGIN  (* Verify the status of the *)
                                      (* I/O operation.           *)
    IF LASTPL<PLANT^.PNUM THEN BEGIN
      SEEK(PLANT,0);            (* Update record 0 if a new record *)
      PLANT^.PNUM:=LASTPL+1;    (* was added.                      *)
      PLANT^.PBNOD:=BASE;
      PLANT^.PFAM:='P';
      PUT(PLANT);
      IF BN>0 THEN BEGIN        (* Update the branch value of the  *)
        SEEK(PLANT,BN);         (* node receiving the new record.  *)
        GET(PLANT);
        IF BRANCH='L' THEN PLANT^.PLNOD:=TRANSF(DUM[7])
        ELSE IF BRANCH='H' THEN PLANT^.PHNOD:=TRANSF(DUM[7]);
        SEEK(PLANT,BN);
        PUT(PLANT);
      END;
      LASTPL:=LASTPL+1;
    END;
  END;
END;  (* STOREPL *)


PROCEDURE GETPLANT;
(*********************************************)
(*                                           *)
(* READ A RECORD FROM THE HOST-PLANT NAME FILE *)
(*                                           *)
(*********************************************)
VAR I:INTEGER;
BEGIN
```

```
  SEEK(PLANT,TRANSF(DUM[7]));  (* Find and read the record. *)
  GET(PLANT);
  IF IOCHECK('PLANT FILE') THEN BEGIN
    WITH PLANT^ DO BEGIN
      SYN:=PSYN;          (* Transfer the content of the record  *)
      SIM:=PSIM;          (* to the working variable.            *)
      LN:=PLNOD;
      HN:=PHNOD;
      BN:=PBNOD;
      DUM[9]:=PGENUS;
      DUM[8]:=PFAM;
      DUM[10]:=PSPEC;
      DUM[11]:=PVAR;
      DUM[12]:=PDESC;
    END;
    FOR I:=7 TO 12 DO BEGIN              (* Print the record on  *)
      GOTOXY(13,LINES[I]);CRT(ERASEOL); (* the screen.          *)
      WRITE(DUM[I]);
    END;
    IF PLANT^.PSIM THEN BEGIN  (* If the host-plant name is a synonym. *)
      REPEAT
        SEEK(PLANT,PLANT^.PBNOD);  (* Find and print the correct *)
        GET(PLANT);                (* name.                      *)
      UNTIL NOT (PLANT^.PSIM);
      GOTOXY(25,LINES[7]);
      WRITE(CHR(15),'Synonym of #',PLANT^.PNUM,CHR(14));
      SEEK(PLANT,TRANSF(DUM[7]));  (* Reload the record.         *)
      GET(PLANT);                  (* The program uses the record*)
    END;                           (* variables to verify if the *)
  END;                             (* name was corrected.        *)
END;  (* GETPLANT *)


PROCEDURE GETCOLL;
(*********************************************)
(*                                           *)
(* READ A RECORD FROM THE COLLECTION DATA FILE *)
(*                                           *)
(*********************************************)
VAR I:INTEGER;
BEGIN
  WITH COLL^ DO BEGIN
    SEEK(COLL,TRANSF(DUM[1])-CCONST);  (* Find and read the record.*)
    GET(COLL);
    IF IOCHECK('COLLECTION FILE') THEN BEGIN
      DUM[2]:=DATE;
      DUM[3]:=CITY;        (* Transfer the content of the record  *)
      DUM[4]:=STATE;       (* to the working variable.            *)
      DUM[5]:=COUNTRY;
      DUM[6]:=COLLECT;
      STR(PNUM,DUM[7]);
      DUM[13]:=PCOMF;
```

```
      DUM[14]:=LOCAT;
      DUM[15]:=COLONY;
      DUM[16]:=COLOR;
      DUM[17]:=CCOM;
      FOR I:=1 TO 6 DO BEGIN                (* Print the record on  *)
        GOTOXY(13,LINES[I]);CRT(ERASEOL); (* the screen.          *)
        WRITE(DUM[I]);
      END;
      FOR I:=13 TO 17 DO BEGIN
        GOTOXY(13,LINES[I]);CRT(ERASEOL);
        WRITE(DUM[I]);
      END;
      CRT(ERASEOS);
    END;
  END;
END;  (* GETCOLL *)


PROCEDURE FINDHOST;
(*********************************************)
(*                                           *)
(* SEARCH A NAME IN THE HOST-PLANT NAME FILE *)
(* AND CORRECT OR UPDATE THE BINARY TREE     *)
(*                                           *)
(*********************************************)
VAR PROX,RPROX,      (* Difference and record number of the name *)
                     (* that match best the working name.        *)
    P:INTEGER;       (* Difference between the working name and  *)
                     (* the record being compared.               *)
    NPROX:STRING;    (* Closest name.                            *)
    ZZ:BOOLEAN;      (* True if the working name is not in file. *)
    REC:INTEGER;     (* Record number of the name being compared *)
                     (* to the working name.                     *)
    PL,DU:STRING;    (* Name being compared and working name     *)
                     (* respectively.                            *)


PROCEDURE SCRATCH;
(*************************************)
(*                                   *)
(* REMOVE A NODE FROM THE BINARY TREE *)
(*                                   *)
(*************************************)
BEGIN
  WITH PLANT^ DO BEGIN
    IF LN<>0 THEN BEGIN          (* If the low branch is not empty. *)
      IF HN<>0 THEN BEGIN        (* If the high branch is not empty.*)
        SEEK(PLANT,HN);          (* Read the high branch record. *)
        GET(PLANT);
        IF PLNOD<>0 THEN BEGIN   (* From the high branch record of  *)
          REPEAT                 (* the working record, follow the   *)
            SEEK(PLANT,PLNOD);   (* tree and find an empty low branch*)
```

```
          GET(PLANT);
        UNTIL PLNOD=0;
      END;
      PLNOD:=LN;                (* Transfer the part of the tree    *)
      SEEK(PLANT,PNUM);         (* beginning at the low branch of   *)
      PUT(PLANT);               (* the working record, to the empty *)
      SEEK(PLANT,LN);           (* low branch found above.          *)
      LN:=PNUM;
      GET(PLANT);
      PBNOD:=LN;
      SEEK(PLANT,PNUM);
      PUT(PLANT);
    END
    ELSE HN:=LN;                (* Same tranfer than above but when *)
  END;                          (* the high branch is empty.        *)
  IF BN<>0 THEN BEGIN     (* If the working record is not the base.*)
    SEEK(PLANT,BN);       (* Find and read the lower node.         *)
    GET(PLANT);
    IF SIM THEN PSYN:=SYN  (* If the working record is a synonym, *)
                           (* remove it from the synonyms chain.  *)
    ELSE IF PLNOD<>TRANSF(DUM[7]) THEN   (* Change the value of   *)
            PHNOD:=HN      (* the branch occupied by the working  *)
         ELSE PLNOD:=HN;   (* record, in the record one node lower*)
    SEEK(PLANT,BN);
    PUT(PLANT);
    END
  ELSE BEGIN                (* If the working record is the base.  *)
    BASE:=HN;
    SEEK(PLANT,0);          (* Change the base value and update the*)
    GET(PLANT);             (* record 0.                           *)
    PLANT^.PBNOD:=BASE;
    SEEK(PLANT,0);
    PUT(PLANT);
  END;
  IF HN<>0 THEN BEGIN       (* Update the node above the working   *)
    SEEK(PLANT,HN);         (* record.                             *)
    GET(PLANT);             (* => remove the working record from   *)
    PBNOD:=BN;              (* the tree.                           *)
    SEEK(PLANT,HN);
    PUT(PLANT);
  END;
  IF SYN<>0 THEN BEGIN      (* Same updating as above but within *)
    SEEK(PLANT,SYN);        (* a synonyms chain.                 *)
    GET(PLANT);
    PBNOD:=BN;
    SEEK(PLANT,SYN);
    PUT(PLANT);
  END;
 END;
END;  (* SCRATCH *)
```

```
PROCEDURE SYNON;
(********************)
(*                  *)
(* CREATE A SYNONYM *)
(*                  *)
(********************)
VAR I:INTEGER;
BEGIN
WITH PLANT^ DO BEGIN
  GOTOXY(0,23);CRT(ERASEOS);
  WRITE(CHR(15),PGENUS,' ',PSPEC,' ',PVAR,' in file,=#',REC,CHR(14));
  IF (SEL IN ['H','h']) AND ((BN<>0) OR (LN<>0) OR (HN<>0)
                    OR (SIM=TRUE)) THEN BEGIN
      (* Continue if correcting the host-plant name file and if the *)
      (* record is not empty.                                       *)
    GOTOXY(0,22);
    WRITE(CHR(15),'Do you want to:create a S)ynonym, E)rase the record,',
          ' or K)eep it unchanged ?',CHR(14),' ');
    WW:=GETCHAR(['S','E','K','s','e','k']);
    CASE WW OF
        'E','e':BEGIN    (* Erase the record.  Assign a null value to *)
                         (* each of the variables and remove the      *)
                         (* record from the tree.                     *)
                  SCRATCH;
                  SEEK(PLANT,TRANSF(DUM[7]));
                  GET(PLANT);
                  PFAM:='';PGENUS:='';
                  PSPEC:='';PVAR:='';PDESC:='';
                  PBNOD:=0;PLNOD:=0;PHNOD:=0;
                  PSYN:=0;PSIM:=FALSE;
                  SEEK(PLANT,PNUM);
                  PUT(PLANT);
                END;
        'S','s':BEGIN   (* Create a synonym. *)
                  SCRATCH;   (* Remove the record from the tree. *)
                  SEEK(PLANT,REC);  (* Find and read the record       *)
                  GET(PLANT);       (* containing the correct name.   *)
                  SYN:=0;
                  IF PSYN<>0 THEN BEGIN  (* If the record already has *)
                    REPEAT               (* synonym(s).               *)
                      SEEK(PLANT,PSYN);  (* Find where to insert the  *)
                                         (* working record. The       *)
                                         (* synonyms chain is arranged*)
                                         (* in decreasing alphabetical*)
                                         (* order.                    *)
                      GET(PLANT);
                      REC:=PNUM;
                    UNTIL ((PSYN=0) OR (CONCAT(PGENUS,PSPEC,PVAR)<DU));
                    IF (CONCAT(PGENUS,PSPEC,PVAR))<DU THEN BEGIN
                       (* If the working record have to be inserted  *)
                       (* within a synonyms chain.                   *)
                      REC:=PBNOD;
```

```
                    PBNOD:=TRANSF(DUM[7]);
                    SYN:=PNUM;
                    SEEK(PLANT,PNUM);
                    PUT(PLANT);
                  END;
                END;
                SEEK(PLANT,REC);              (* Change the value of *)
                GET(PLANT);                   (* the synonym branch  *)
                BN:=PNUM;                     (* of the node before  *)
                PSYN:=TRANSF(DUM[7]);         (* the working record. *)
                SEEK(PLANT,REC);
                PUT(PLANT);
                SEEK(PLANT,TRANSF(DUM[7]));  (* Update the variables*)
                GET(PLANT);                   (* of the working      *)
                PSIM:=TRUE;                   (* record and save.    *)
                PSYN:=SYN;
                PBNOD:=BN;
                PLNOD:=0;
                PHNOD:=0;
                SEEK(PLANT,PNUM);
                PUT(PLANT);
              END;
    END;
  END
  ELSE BEGIN   (* If not correcting the host-plant name file. *)
    STR(PNUM,DUM[7]);  (* Change and print the reference number. *)
    GOTOXY(13,LINES[7]);CRT(ERASEOL);
    WRITE(DUM[7]);
  END;
  NEWREC:=FALSE;
  BRANCH:=' ';
  EXIT(FINDHOST);
  END;
END;  (* SYNON *)


BEGIN  (* FINDHOST *)
  IF LASTPL<1 THEN EXIT (FINDHOST);  (* If the host-plant name file *)
  ZZ:=FALSE;                         (* is empty or contains only   *)
  WITH PLANT^ DO BEGIN               (* one record.                 *)
    IF NOT (SEL IN ['H','h']) THEN BEGIN  (* If not correcting the  *)
      SIM:=FALSE;                         (* host-plant name file.  *)
      SYN:=0;LN:=0;HN:=0;BN:=0;
    END;
    REC:=BASE;PROX:=70;RPROX:=0;NPROX:='';
    REPEAT
      SEEK(PLANT,REC);  (* Start at the base of the tree. *)
      GET(PLANT);
      IF IOCHECK('PLANT FILE') THEN BEGIN
        PL:=CONCAT(PGENUS,PSPEC,PVAR);       (* Compare the record name*)
        DU:=CONCAT(DUM[9],DUM[10],DUM[11]); (* to the working name and*)
                                             (* retain it if it is the *)
```

```
                                        (* closest so far.        *)
P:=SIMIL(PGENUS,DUM[9])+SIMIL(PSPEC,DUM[10])+SIMIL(PVAR,DUM[11]);
IF P<=PROX THEN BEGIN
  PROX:=P;
  RPROX:=REC;
  NPROX:=CONCAT(PGENUS,' ',PSPEC,' ',PVAR);
END;
IF PL=DU THEN SYNON  (* Create a synonym. *)
ELSE BEGIN
  IF DU<PL THEN BEGIN          (* According to the alphabetical*)
    IF PLNOD>0 THEN REC:=PLNOD (* value of the working name,  *)
    ELSE BEGIN                 (* determine the branch to:     *)
      BRANCH:='L';             (*      follow (if not empty),  *)
      ZZ:=TRUE;                (*      add the working name to *)
    END;                       (*            (if empty).       *)
  END
  ELSE BEGIN
    IF PHNOD>0 THEN REC:=PHNOD
    ELSE BEGIN
      BRANCH:='H';
      ZZ:=TRUE;
    END;
  END;
  IF ZZ THEN BEGIN
    GOTOXY(13,LINES[7]);CRT(ERASEOL);
    WRITE(DUM[7]);     (* Print the reference number. *)
    GOTOXY(25,LINES[7]);
    WRITE(CHR(15),'New host-plant name',CHR(14));
    GOTOXY(0,23);CRT(ERASEOS);
    WRITE(CHR(15),'Closest host-plant: ',RPROX,' ',NPROX,CHR(14));
    IF NOT (SEL IN ['h','H']) THEN BEGIN  (* If not correcting *)
      BN:=REC;                      (* the host-plant name file. *)
      EXIT(FINDHOST);
    END
    ELSE BEGIN  (* If correcting the host-plant name file. *)
      IF REC=TRANSF(DUM[7]) THEN EXIT(FINDHOST);
      (* Exit if the corrected working record would have to *)
      (* be added after itself.                             *)
      GOTOXY(0,22);
      CRT(ERASEOL);
      WRITE(CHR(15),'Save the changes ?',CHR(14),' ');
      IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
        WITH PLANT^ DO BEGIN  (* Save the working record and *)
                              (* correct its position within *)
                              (* the binary tree.            *)
          IF (BN<>0) OR (LN<>0) OR (HN<>0) OR (SYN<>0) THEN SCRATCH;
          (* Remove the working record from the tree if it was *)
          (* not an empty record.                              *)
          BN:=REC;
          LN:=0;
          HN:=0;
          SYN:=0;
```

```
                SIM:=FALSE;
                STOREPL;
                SEEK(PLANT,REC);
                GET(PLANT);
                IF BRANCH='L' THEN PLNOD:=TRANSF(DUM[7])
                ELSE PHNOD:=TRANSF(DUM[7]);
                SEEK(PLANT,REC);
                PUT(PLANT);
              END;
            END;
            NEWREC:=FALSE;
            EXIT(FINDHOST);
          END;
        END;
      END;
    END;
  UNTIL FALSE;
 END;
END;  (* FINDHOST *)


PROCEDURE CAPITAL(K:INTEGER);
(*************************************************)
(*                                               *)
(* CHANGE THE FIRST CHARACTER OF A STRING FOR THE *)
(* CORRESPONDING UPPER CASE CHARACTER AND THE     *)
(* OTHER CHARACTERS FOR THE CORRESPONDING LOWER   *)
(* CASE CHARACTERS.                               *)
(*                                               *)
(*************************************************)
VAR I:INTEGER;
BEGIN
  IF LENGTH(DUM[K])>0 THEN BEGIN
    IF DUM[K][1] IN ['a'..'z'] THEN
                  DUM[K][1]:=CHR((ORD(DUM[K][1]))-32);
    IF LENGTH(DUM[K])>1 THEN
      FOR I:=2 TO LENGTH(DUM[K]) DO BEGIN
        IF DUM[K][I] IN ['A'..'Z'] THEN
                  DUM[K][I]:=CHR(32+(ORD(DUM[K][I])));
      END;
  END;
  GOTOXY(13,LINES[K]);CRT(ERASEOL);  (* Print the string. *)
  WRITE(DUM[K]);
END;  (* CAPITAL *)
```

```
PROCEDURE CORPLANT;
(*********************************)
(*                               *)
(* CORRECT THE HOST-PLANT NAME FILE *)
(*                               *)
(*********************************)
VAR YES:BOOLEAN;
    I:INTEGER;
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(25,0);  (* Print the title. *)
  WRITE(CHR(15),'CORRECTING THE HOST-PLANT NAME FILE',CHR(14));
  REPEAT
  NEWREC:=TRUE;
    REPEAT
      GETSTRING(7,7);  (* Read from the keyboard, the reference *)
                       (* number of the record to be corrected, *)
                       (* and verify if it is within range.     *)
    UNTIL (TRANSF(DUM[7])>0) AND (LASTPL>=TRANSF(DUM[7]));
    GETPLANT;  (* Read the record from the file. *)
    IF IORES=0 THEN BEGIN
      GETSTRING(8,12);  (* Read the corrections. *)
      CAPITAL(8);
      CAPITAL(9);
      IF (PLANT^.PGENUS<>DUM[9]) OR (PLANT^.PSPEC<>DUM[10]) OR
         (PLANT^.PVAR<>DUM[11]) THEN FINDHOST
         (* If the host-plant genus, species or variety was     *)
         (* changed, search for the corrected name in the file. *)
      ELSE IF PLANT^.PSIM THEN BEGIN
         (* If the working record is a synonym, the record can  *)
         (* be: corrected or removed from the synonyms list and *)
         (* added to the binary tree.                           *)
        GOTOXY(0,22);CRT(ERASEOL);
        WRITE(CHR(15),'Keep it as a synonym ?',CHR(14),' ');
        IF GETCHAR(['Y','N','y','n']) IN ['N','n'] THEN FINDHOST;
      END;
      IF NEWREC THEN BEGIN  (* If the procedure 'FINDHOST' was  *)
                            (* not executed.                    *)
        GOTOXY(0,22);CRT(ERASEOL);
        WRITE(CHR(15),'Save the changes ?',CHR(14),' ');
        IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN STOREPL;
      END;
      GOTOXY(0,22);CRT(ERASEOS);
      WRITE(CHR(15),'Another record ??',CHR(14),' ');
      YES:=GETCHAR(['Y','y','N','n']) in ['n','N'];
    END
    ELSE YES:=TRUE;
  UNTIL YES;
END;  (* CORPLANT *)
```

```
PROCEDURE UPDPLANT;
(*********************************)
(*                               *)
(* UPDATE THE HOST-PLANT NAME FILE *)
(*                               *)
(*********************************)
VAR YES:BOOLEAN;
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(24,0);  (* Print the title. *)
  WRITE(CHR(15),'UPDATING THE HOST-PLANT NAME FILE',CHR(14));
  REPEAT
    IF LASTPL+1=MAXINT THEN BEGIN
      (* Exit if the reference number of the name to be added *)
      (* is 32767.                                            *)
      GOTOXY(0,22);CRT(ERASEOS);
      WRITE(CHR(15),'Largest acceptable host-plant number reached.');
      WRITE(' Open a new file.',CHR(14));
      GOTOXY(0,23);
      WRITE(CHR(15),'Press spacebar to continue.',CHR(14));
      WW:=GETCHAR([' ']);
      EXIT(UPDPLANT);
    END;
    NEWREC:=TRUE;
    STR(LASTPL+1,DUM[7]);  (* Print the reference number. *)
    GOTOXY(13,LINES[7]);CRT(ERASEOL);
    WRITE(DUM[7]);
    GETSTRING(8,12);       (* Read the record from the keyboard. *)
    CAPITAL(9);
    CAPITAL(8);
    FINDHOST;
    IF NEWREC THEN BEGIN  (* If it is a new name. *)
      GOTOXY(0,22);CRT(ERASEOL);
      WRITE(CHR(15),'Save the record ??',CHR(14),' ');
      IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN STOREPL;
    END;
    IF IORES=0 THEN BEGIN
      GOTOXY(0,22);CRT(ERASEOS);
      WRITE(CHR(15),'Another record ??',CHR(14),' ');
      YES:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
    END
    ELSE YES:=TRUE;
  UNTIL YES;
END;  (* UPDPLANT *)
```

```
PROCEDURE CORRCOLL;
(*********************************)
(*                               *)
(* CORRECT THE COLLECTION DATA FILE *)
(*                               *)
(*********************************)
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(23,0);            (* Print the title. *)
  WRITE(CHR(15),'CORRECTING THE COLLECTION DATA FILE',CHR(14));
  REPEAT
    NEWREC:=FALSE;
    REPEAT
      GETSTRING(1,1);  (* Read the reference number from the keyboard,*)
                       (* and verify if it is within range.          *)
    UNTIL ((TRANSF(DUM[1])-CCONST)>0) AND ((TRANSF(DUM[1])-CCONST)<=LASTCOL);
    GETCOLL;           (* Read the collection data record from the file. *)
    IF IORES<>0 THEN BYEBYE;
    GETPLANT;          (* If the I/O operation was successful, read   *)
                       (* the host-plant name record.                 *)
    IF IORES<>0 THEN BYEBYE;
    GETSTRING(2,17);  (* Read the corrections from the keyboard. *)
    CAPITAL(8);
    CAPITAL(9);
    IF (PLANT^.PGENUS<>DUM[9]) OR (PLANT^.PSPEC<>DUM[10])
       OR (PLANT^.PVAR<>DUM[11]) THEN BEGIN
       (* If the host-plant genus, species or variety was changed, *)
       (* search for the new name in the host-plant name file.     *)
       NEWREC:=TRUE;
       FINDHOST;
    END
    ELSE IF (COLL^.PNUM<>TRANSF(DUM[7])) THEN
           (* If the host-plant reference number was changed, read *)
           (* the new record from the host-plant name file.        *)
           IF (TRANSF(DUM[7])>0) AND
                       (TRANSF(DUM[7])<=LASTPL) THEN GETPLANT
           ELSE BEGIN  (* If the host-plant reference number was   *)
                       (* out of range, restore the last number.   *)
             STR(PLANT^.PNUM,DUM[7]);
             GOTOXY(13,LINES[7]);CRT(ERASEOL);
             WRITE(DUM[7]);
             WRITE(' ',CHR(15),'Out of range, ',
                    'previous number restored.',CHR(14));
           END
         ELSE BEGIN  (* If the host-plant family was changed,     *)
                     (* display an error message.                 *)
             IF PLANT^.PFAM<>DUM[8] THEN BEGIN
               DUM[8]:=PLANT^.PFAM;
               GOTOXY(13,LINES[8]);CRT(ERASEOL);
               WRITE(DUM[8]);
               WRITE('      ',CHR(15),'To correct use selection ',
                     'H in the main menu.',CHR(14));
```

```
          END;
          IF PLANT^.PDESC<>DUM[12] THEN BEGIN
                    (* If the host-plant describer was changed,  *)
                    (* display an error message.                 *)
            DUM[12]:=PLANT^.PDESC;
            GOTOXY(13,LINES[12]);CRT(ERASEOL);
            WRITE(DUM[12]);
            WRITE('     ',CHR(15),'To correct use selection ',
                  'H in the main menu.',CHR(14));
          END;
        END;
    GOTOXY(0,22);CRT(ERASEOL);
    WRITE(CHR(15),'Save the changes??',CHR(14),' ');
    IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
      IF NEWREC THEN BEGIN    (* Save the host-plant record if it *)
        STR(LASTPL+1,DUM[7]); (* is a new name.                   *)
        GOTOXY(13,LINES[7]);
        WRITE(DUM[7]);
        CRT(ERASEOL);
        STOREPL;
      END;
      IF IORES<>0 THEN BYEBYE
      ELSE STORECOL;            (* Save the collection data record *)
                                (* if the host-plant record was    *)
                                (* properly saved.                 *)
    END;
    GOTOXY(0,22);CRT(ERASEOS);
    WRITE(CHR(15),'Another record??',CHR(14),' ');
  UNTIL GETCHAR(['Y','N','y','n']) IN ['N','n'];
  BYEBYE;
END;  (* CORRCOLL *)


PROCEDURE UPDCOLL;
(*********************************)
(*                               *)
(* UPDATE THE COLLECTION DATA FILE *)
(*                               *)
(*********************************)
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(23,0);    (* Print the title. *)
  WRITE(CHR(15),'UPDATING THE COLLECTION DATA FILE',CHR(14));
  REPEAT
    IF (LASTCOL+CCONST+1)=MAXINT THEN BEGIN
    (* Exit if the number of records in the file is 32766. *)
      GOTOXY(0,22);CRT(ERASEOS);
      WRITE(CHR(15),'Largest acceptable collection number reached.',
            '  Open a new file.',CHR(14));
      GOTOXY(0,23);
      WRITE('Press spacebar to continue.',CHR(14));
      WW:=GETCHAR([' ']);
```

```
      BYEBYE;
    END;
    NEWREC:=FALSE;
    IF LASTCOL=0 THEN BEGIN  (* If working with the first record  *)
      REPEAT                 (* of a new file, read the reference *)
        GETSTRING(1,1);      (* number.                           *)
      UNTIL TRANSF(DUM[1])>0;
    END
    ELSE BEGIN                     (* If not the first record of a *)
    STR(LASTCOL+CCONST+1,DUM[1]); (* new file, print the reference*)
    GOTOXY(13,LINES[1]);CRT(ERASEOL);(* number.                    *)
    WRITE(DUM[1]);
    END;
    GETSTRING(2,17);  (* Read the collection data record from the *)
    CAPITAL(8);       (* Keyboard.                                *)
    CAPITAL(9);
    IF (TRANSF(DUM[7])>0) AND
                  (TRANSF(DUM[7])<=LASTPL) THEN GETPLANT
         (* If the host-plant reference number is within range,   *)
         (* read the record from the host-plant name file.        *)
    ELSE BEGIN        (* If the host-plant reference number is    *)
      NEWREC:=TRUE;   (* absent or out of range, search for the   *)
      FINDHOST;       (* name in the host-plant name file.        *)
    END;
    GOTOXY(0,22);CRT(ERASEOL);
    WRITE(CHR(15),'Save the record??',CHR(14),' ');
    IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
      IF NEWREC THEN BEGIN    (* Save the host-plant record if it *)
        STR(LASTPL+1,DUM[7]); (* is a new name.                   *)
        GOTOXY(13,LINES[7]);
        WRITE(DUM[7]);
        CRT(ERASEOL);
        STOREPL;
      END;
      IF IORES=0 THEN STORECOL  (* If the I/O operation was        *)
      ELSE BYEBYE;              (* successful, save the collection *)
                                (* data record.                    *)
    END;
    GOTOXY(0,22);CRT(ERASEOS);
    WRITE(CHR(15),'Another record??',CHR(14),' ');
  UNTIL GETCHAR(['Y','N','y','n']) IN ['N','n'];
  BYEBYE;
END;  (* UPDCOLL *)


BEGIN (*PROGRAM*)
  GETCVAL(DUM[1]);  (* Get the selection code from 'TITLE'. *)
  SEL:=DUM[1][1];
  GETCRTINFO;
  SETCHAIN('COLLECT:TITLE');  (* Select the next program to be executed*)
  RESET(FNAMES,'COLLECT:FILEINFO');  (* Read the last file names used. *)
  INIT;          (* Initialization steps. *)
```

```
  CASE SEL OF
  'G','g':BEGIN            (* Update the host-plant name file. *)
            DISP_C;
            UPDPLANT;
          END;
  'H','h':BEGIN            (* Correct the host-plant name file. *)
            DISP_C;
            GOTOXY(0,22);  (* Print the largest host-plant reference *)
                           (* number of the file.                    *)
            WRITE(CHR(15),'Largest host-plant number is ',LASTPL,CHR(14));
            CORPLANT;
          END;
  'A','a':BEGIN            (* Update the collection data file. *)
            DISP_B;
            DISP_C;
            UPDCOLL;
          END;
  'B','b':BEGIN            (* Correct the collection data file. *)
            DISP_B;
            DISP_C;
            GOTOXY(0,22); (* Print the smallest and the largest       *)
                          (* collection reference number of the file. *)
            WRITE(CHR(15),'Collection numbers range from ',CCONST+1,
                     ' to ',CCONST+LASTCOL,CHR(14));
            CORRCOLL;
          END;
  END;
  BYEBYE
END.  (* HOST *)
```

```
PROGRAM SLIDEAPHID;
(*****************************************)
(*                                       *)
(* CORRECT OR UPDATE THE SLIDE DATA FILE OR *)
(* THE APHID NAME FILE                   *)
(*                                       *)
(*****************************************)
USES CHAINSTUFF;
TYPE SETOFCHAR=SET OF CHAR;
    CRTCOMMAND= (ERASEOS,ERASEOL,UP,DOWN,RIGHT,LEFT,LEADIN);
    LONG=INTEGER[6];
    SLONG=STRING[6];
VAR SLIDE:FILE OF RECORD
            SNUM:INTEGER[6];
            CNUM:INTEGER;
            ANUM:INTEGER;
            ACOMF:  STRING[1];  (* See Table 5 for a description *)
            MORPH: STRING[12];  (* of the variables.             *)
            IDENT: STRING[20];
            MANY:STRING[2];
            SCOM:STRING[65];
          END;
   APHID:FILE OF PACKED RECORD
           ANUM:INTEGER;
           AGENUS:STRING[20];
           ASBGN:STRING[20];    (* See Table 2 for a description *)
           ASPEC:STRING[20];    (* of the variables.             *)
           ASBSP:STRING[20];
           ADESC:STRING[20];
           ASIM:BOOLEAN;
           ASYN:INTEGER;
           AHNOD:INTEGER;
           ALNOD:INTEGER;
           ABNOD:INTEGER;
         END;
    FNAMES:FILE OF RECORD
             CFILE,PFILE,SFILE,AFILE:STRING;
           END;
    DUM:ARRAY[1..13] OF  STRING;          (* The variables are        *)
    LINES:ARRAY[1..13] OF INTEGER;        (* described in the program *)
    DUML:ARRAY[1..13] OF INTEGER;         (* 'HOST'.                  *)
    NEWREC,SIM:BOOLEAN;
    OKSET:ARRAY[1..13] OF SETOFCHAR;
    SCONST:LONG;                 (* Slide reference number (minus one) of *)
                                 (* the first record in the file.         *)
    LASTSL,LASTAP,               (* Number of records in the slide data   *)
                                 (* file and in the aphid name file.      *)
    I,IORES:INTEGER;
    LN,HN,BN,SYN,BASE:INTEGER;
    SEL,BRANCH,WW:CHAR;
    CRTINFO: PACKED ARRAY[CRTCOMMAND] OF CHAR;
    PREFIXED:ARRAY[CRTCOMMAND] OF BOOLEAN;
```

```
(*$I-*)


PROCEDURE GETCRTINFO;
(*****************************************)
(*                                       *)
(* READ THE SCREEN CONTROL CHARACTERS FROM *)
(* THE SYSTEM.MISCINFO FILE              *)
(*                                       *)
(*****************************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE CRT(C:CRTCOMMAND);
(************************)
(*                      *)
(*   SCREEN COMMANDS    *)
(*                      *)
(************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE BYEBYE;
(*************************************)
(*                                   *)
(* UPDATE FILEINFO AND CLOSE THE FILES *)
(*                                   *)
(*************************************)
BEGIN
  SEEK(FNAMES,0);
  PUT(FNAMES);
  CLOSE(SLIDE,LOCK);
  CLOSE(APHID,LOCK);
  CLOSE(FNAMES,LOCK);
  EXIT(PROGRAM);
END;  (* BYEBYE *)


PROCEDURE TRLONG(VAR WW:SLONG;VAR TEMP:LONG);
(*****************************************************)
(*                                                   *)
(* CONVERT THE VALUE OF A STRING INTO A LONG INTEGER *)
(*                                                   *)
(*****************************************************)
VAR I,J:INTEGER;
    VAL:INTEGER[6];
BEGIN
  TEMP:=0;VAL:=0;
  IF (WW='') OR (WW>'999999') THEN EXIT(TRLONG);
        (* Return 0 if the value of the string cannot be *)
```

```
        (* converted into a long integer.                    *)
  FOR I:=1 TO LENGTH(WW) DO BEGIN
    VAL:=ORD(WW[I])-48;
    IF I<LENGTH(WW) THEN
      FOR J:=1 TO LENGTH(WW)-I DO
        VAL:=VAL*10;
    TEMP:=TEMP+VAL;
  END;
END;  (* TRLONG *)


FUNCTION TRANSF(Z:STRING):INTEGER;
(*************************************************)
(*                                               *)
(* CONVERT THE VALUE OF A STRING INTO AN INTEGER *)
(*                                               *)
(*************************************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE DISP_D;
(*********************************************)
(*                                           *)
(* DISPLAY THE INVERSE VIDEO LEFT COLUMN FOR *)
(* THE SLIDE DATA VARIABLES                  *)
(*                                           *)
(*********************************************)
BEGIN
   WRITE(CHR(15));
   GOTOXY(0,LINES[1]);WRITE('SLIDE NUMBER:');
   GOTOXY(0,LINES[2]);WRITE('COLLECTION #:');
   GOTOXY(0,LINES[9]);WRITE('POSITIVE ID?:');
   GOTOXY(0,LINES[10]);WRITE('MORPHS:......');
   GOTOXY(0,LINES[11]);WRITE('DETERMINATOR:');
   GOTOXY(0,LINES[12]);WRITE('SPECIMENS, #:');
   GOTOXY(0,LINES[13]);WRITE('COMMENTS:....');
   WRITE(CHR(14));
END;  (* DISP_D *)


PROCEDURE DISP_E;
(*********************************************)
(*                                           *)
(* DISPLAY THE INVERSE VIDEO LEFT COLUMN FOR *)
(* THE APHID NAME VARIABLES                  *)
(*                                           *)
(*********************************************)
BEGIN
   WRITE(CHR(15));
   GOTOXY(0,LINES[3]);WRITE('APHID NUMBER:');
   GOTOXY(0,LINES[4]);WRITE('APHID GENUS: ');
```

```
   GOTOXY(0,LINES[5]);WRITE('APHID SUBGEN:');
   GOTOXY(0,LINES[6]);WRITE('APHID SP.:...');
   GOTOXY(0,LINES[7]);WRITE('APHID SUBSP.:');
   GOTOXY(0,LINES[8]);WRITE('APHID DESCR.:');
   WRITE(CHR(14));
END;  (* DISP_E *)


FUNCTION GETCHAR(OK:SETOFCHAR):CHAR;
(*************************************)
(*                                   *)
(* GET A CHARACTER FROM THE KEYBOARD *)
(*                                   *)
(*************************************)

(* See the program 'TITLE' for the listing and comments. *)


PROCEDURE GETSTRING(S,E:INTEGER);
(*******************************)
(*                             *)
(*       APHID EDITOR          *)
(*                             *)
(*******************************)

(* See the program 'HOST' for the listing and comments. *)


FUNCTION SIMIL (A,B:STRING):INTEGER;
(*********************************************)
(*                                           *)
(* COMPUTE THE DIFFERENCE (NUMBER OF MATCHLESS *)
(* CHARACTERS) BETWEEN TWO STRINGS             *)
(*                                           *)
(*********************************************)

(* See the program 'HOST' for the listing and comments. *)


FUNCTION IOCHECK(F:STRING):BOOLEAN;
(************************************)
(*                                  *)
(* VERIFY THE STATUS OF THE LAST I/O *)
(* OPERATION AND PRINT AN ERROR      *)
(* MESSAGE IF THE OPERATION WAS NOT  *)
(* SUCCESSFUL                        *)
(*                                  *)
(************************************)

(* See the program 'HOST' for the listing and comments. *)
```

```
PROCEDURE INIT;
(************************)
(*                      *)
(* INITIALIZATION STEPS *)
(*                      *)
(************************)
VAR I:INTEGER;


PROCEDURE INITSL;
(*********************************************)
(*                                           *)
(* INITIALIZATION OF THE VARIABLES NEEDED TO *)
(* WORK ON THE SLIDE DATA FILE               *)
(*                                           *)
(*********************************************)
BEGIN
  DUML[1]:=6;
  DUML[2]:=5;
  DUML[9]:=1;
  DUML[10]:=12;
  DUML[11]:=20;
  DUML[12]:=2;
  DUML[13]:=65;
  OKSET[1]:=['0'..'9'];
  OKSET[2]:=['0'..'9'];
  OKSET[9]:=['?'];
  OKSET[10]:=['A'..'Z','a'..'z',',','.',' '];
  OKSET[11]:=OKSET[10]+['(',')'];
  OKSET[12]:=OKSET[1];
  OKSET[13]:=[' '..'#','%'..'.','0'..'Z','a'..'z'];
END;  (* INITSL *)


PROCEDURE INITAP;
(*********************************************)
(*                                           *)
(* INITIALIZATION OF THE VARIABLES NEEDED TO *)
(* WORK ON THE APHID NAME FILE               *)
(*                                           *)
(*********************************************)
BEGIN
  DUML[3]:=5;
  FOR I:=4 TO 8 DO DUML[I]:=20;
  OKSET[3]:=['0'..'9'];
  OKSET[4]:=['A'..'Z','a'..'z',' '];
  OKSET[5]:=OKSET[4];
  OKSET[6]:=['a'..'z','-',' '];
  OKSET[7]:=OKSET[4];
  OKSET[8]:=OKSET[7]+['(',')','.'];
END;  (* INITAP *)
```

```
PROCEDURE OPEN (A:CHAR;L:INTEGER);
(*******************************)
(*                             *)
(*     NAME AND OPEN FILES     *)
(*                             *)
(*******************************)
(* The record 0 of each file contains information about the file. *)
(* For the slide data file:                                       *)
(*      SLIDE^.CNUM=number of records in the file                 *)
(*      SLIDE^.SNUM=first slide reference number (minus one) in   *)
(*                  the file.                                     *)
(*      SLIDE ^.ACOMF=slide data file identification code.        *)
(* For the aphid name file:                                       *)
(*      APHID^.ANUM=number of records in the file                 *)
(*      APHID^.AGENUS=aphid name file identification code.        *)

(* This procedure is the same as the 'OPEN' procedure of the      *)
(* program 'HOST'.  Only the variables names were changed.  Refer *)
(* to 'OPEN' in the program 'HOST' for comments.                  *)

VAR OLD,RES,CH:BOOLEAN;
    CHOI:CHAR;


PROCEDURE OP1;
(*******************************)
(*                             *)
(*   GET THE FULL FILE NAME    *)
(*                             *)
(*******************************)
VAR RENAME:BOOLEAN;
BEGIN
  GOTOXY(0,L);CRT(ERASEOL);
  CASE A OF
    'C':BEGIN
          WRITE(CHR(15),'Slide data file=',CHR(14));
          IF (FNAMES^.SFILE='') OR (CH=TRUE) THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.SFILE,'  ');
            RENAME:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(17,L);CRT(ERASEOL);
            READLN(FNAMES^.SFILE);
          END;
        END;
    'D':BEGIN
          WRITE(CHR(15),'Aphid name file=',CHR(14));
          IF (FNAMES^.AFILE='') OR (CH=TRUE) THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.AFILE,'  ');
```

```
            RENAME:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(17,L);CRT(ERASEOL);
            READLN(FNAMES^.AFILE);
          END;
        END;
  END;
END;  (* OP1 *)


BEGIN  (* OPEN *)
  OLD:=TRUE;
  CH:=FALSE;
  OP1;
  REPEAT
    IF CH=TRUE THEN OP1;
    IF OLD THEN BEGIN
      CASE A OF
        'C':RESET(SLIDE,FNAMES^.SFILE);
        'D':RESET(APHID,FNAMES^.AFILE);
      END;
    END
    ELSE BEGIN
      CASE A OF
        'C':REWRITE(SLIDE,FNAMES^.SFILE);
        'D':REWRITE(APHID,FNAMES^.AFILE);
      END;
    END;
    RES:=IOCHECK(' ');
    IF NOT (IORES IN [5,6,9]) THEN BEGIN
      IF RES=FALSE THEN BEGIN
        GOTOXY(0,L+2);CRT(ERASEOL);
        WRITE(CHR(15),'Choose between: C)hange the file name, N)ew file,',
             ' R)eturn to main menu.',CHR(14),' ');
        CHOI:=GETCHAR(['C','N','R','c','n','r']);
        CASE CHOI OF
          'C','c':CH:=TRUE;
          'N','n':BEGIN
                    CH:=FALSE;
                    OLD:=FALSE;
                  END;
          'R','r':BYEBYE;
        END;
      END
      ELSE BEGIN
        IF OLD THEN BEGIN
          CASE A OF
            'C':BEGIN
                  IF SLIDE^.ACOMF<>'S' THEN BEGIN
                    GOTOXY(0,L+2);CRT(ERASEOL);
                    WRITE(CHR(15),FNAMES^.SFILE,' is not',
```

```
                    ' a slide data file.',CHR(14));
                CH:=TRUE;RES:=FALSE;
                CLOSE(SLIDE,LOCK);
              END
              ELSE BEGIN
                LASTSL:=SLIDE^.CNUM;
                SCONST:=SLIDE^.SNUM;
              END;
            END;
        'D':BEGIN
              IF APHID^.AGENUS<>'A' THEN BEGIN
                GOTOXY(0,L+2);CRT(ERASEOL);
                WRITE(CHR(15),FNAMES^.AFILE,' is not',
                    ' an aphid name file.',CHR(14));
                CH:=TRUE;RES:=FALSE;
                CLOSE(APHID,LOCK);
              END
              ELSE BEGIN
                LASTAP:=APHID^.ANUM;
                BASE:=APHID^.ABNOD;
              END;
            END;
      END;
    END
    ELSE BEGIN
      CASE A OF
        'C':BEGIN
              LASTSL:=0;
              SCONST:=0;
            END;
        'D':BEGIN
              LASTAP:=0;
              BASE:=1;
            END;
      END;
    END;
  END;
END;
UNTIL RES;
END;(* OPEN *)


BEGIN (*INIT*)
  WRITE(CHR(12));
  GOTOXY(30,0);
  WRITE(CHR(15),'INITIALIZATION',CHR(14));
  CASE SEL OF
    'C','D','c','d':BEGIN
                      FOR I:=1 TO 13 DO LINES[I]:=I+4;
                      INITSL;
                      INITAP;
                      OPEN('C',2);
```

```
                    OPEN('D',4);
                  END;
    'E','F','e','f':BEGIN
                    FOR I:=1 TO 6 DO LINES[I+2]:=I+4;
                    INITAP;
                    OPEN('D',2);
                  END;
  END;
  FOR I:=1 TO 13 DO DUM[I]:='';
  WRITE(CHR(12));
  LN:=0;BN:=0;HN:=0;SYN:=0;SIM:=FALSE;
END;  (* INIT *)


PROCEDURE STORESL;
(*****************************************)
(*                                       *)
(* SAVE A RECORD INTO THE SLIDE DATA FILE *)
(*                                       *)
(*****************************************)
BEGIN
  WITH SLIDE^ DO BEGIN
    TRLONG(DUM[1],SNUM);  (* Transfer the values of the working *)
    CNUM:=TRANSF(DUM[2]); (* variable to the record variables.  *)
    ANUM:=TRANSF(DUM[3]);
    ACOMF:=DUM[9];
    MORPH:=DUM[10];
    IDENT:=DUM[11];
    MANY:=DUM[12];
    SCOM:=DUM[13];
  END;
  IF LASTSL=0 THEN SCONST:=SLIDE^.SNUM-1;  (* Initialize SCONST if    *)
  SEEK(SLIDE,TRUNC(SLIDE^.SNUM-SCONST));   (* working on a new file.  *)
  PUT(SLIDE);
  IF IOCHECK('SLIDE FILE') THEN BEGIN      (* Verify the status of    *)
                                           (* the I/O operation.      *)
    IF LASTSL<TRUNC(SLIDE^.SNUM-SCONST) THEN BEGIN
      SEEK(SLIDE,0);       (* Update the record 0, if a new     *)
      LASTSL:=LASTSL+1;    (* record was saved.                 *)
      SLIDE^.CNUM:=LASTSL;
      SLIDE^.SNUM:=SCONST;
      SLIDE^.ACOMF:='S';
      PUT(SLIDE);
    END;
  END;
END;  (* STORESL *)
```

```
PROCEDURE STOREAP;
(*****************************************)
(*                                       *)
(* SAVE A RECORD INTO THE APHID NAME FILE *)
(*                                       *)
(*****************************************)
BEGIN
  WITH APHID^ DO BEGIN
    ANUM:=TRANSF(DUM[3]);  (* Transfer the values of the working *)
    ABNOD:=BN;             (* variable to the record variables.  *)
    ALNOD:=LN;
    AHNOD:=HN;
    ASYN:=SYN;
    ASIM:=SIM;
    AGENUS:=DUM[4];
    ASBGN:=DUM[5];
    ASPEC:=DUM[6];
    ASBSP:=DUM[7];
    ADESC:=DUM[8];
  END;
  SEEK(APHID,TRANSF(DUM[3]));  (* Save the record. *)
  PUT(APHID);
  IF IOCHECK('APHID FILE')THEN BEGIN  (* Verify the status of the   *)
                                      (* I/O operation.             *)
    IF LASTAP<APHID^.ANUM THEN BEGIN
      SEEK(APHID,0);
      APHID^.ANUM:=LASTAP+1;   (* Update the record 0 if a new      *)
      APHID^.ABNOD:=BASE;      (* record was added.                 *)
      APHID^.AGENUS:='A';
      PUT(APHID);
      IF BN>0 THEN BEGIN       (* Update the branch value of the    *)
        SEEK(APHID,BN);        (* node receiving the new record.    *)
        GET(APHID);
        IF BRANCH='L' THEN APHID^.ALNOD:=TRANSF(DUM[3])
        ELSE IF BRANCH='H' THEN APHID^.AHNOD:=TRANSF(DUM[3]);
        SEEK(APHID,BN);
        PUT(APHID);
      END;
      LASTAP:=LASTAP+1;
    END;
  END;
END;(* STOREAP *)


PROCEDURE GETAPHID;
(*****************************************)
(*                                       *)
(* READ A RECORD FROM THE APHID NAME FILE *)
(*                                       *)
(*****************************************)
VAR I:INTEGER;
BEGIN
```

```
SEEK(APHID,TRANSF(DUM[3]));   (* Find and read the record. *)
GET(APHID);
IF IOCHECK('APHID FILE') THEN BEGIN
  WITH APHID^ DO BEGIN
    SYN:=ASYN;
    SIM:=ASIM;   (* Transfer the content of the record to the  *)
    LN:=ALNOD;   (* working variable.                          *)
    HN:=AHNOD;
    BN:=ABNOD;
    DUM[4]:=AGENUS;
    DUM[5]:=ASBGN;
    DUM[6]:=ASPEC;
    DUM[7]:=ASBSP;
    DUM[8]:=ADESC;
  END;
  FOR I:=3 TO 8 DO BEGIN    (* Print the record on the screen. *)
    GOTOXY(13,LINES[I]);CRT(ERASEOL);
    WRITE(DUM[I]);
  END;
  IF APHID^.ASIM THEN BEGIN   (* If it is a synonym. *)
    REPEAT
      SEEK(APHID,APHID^.ABNOD);   (* Find and print the correct name*)
      GET(APHID);
    UNTIL NOT (APHID^.ASIM);
    GOTOXY(25,LINES[3]);
    WRITE(CHR(15),'Synonym of #',APHID^.ANUM,CHR(14));
    SEEK(APHID,TRANSF(DUM[3]));    (* Reload the record.          *)
    GET(APHID);                    (* The program uses the record *)
  END;                             (* variables to verify if the  *)
 END;                              (* name was corrected.         *)
END;  (* GETAPHID *)


PROCEDURE GETSLIDE;
(*****************************************)
(*                                       *)
(* READ A RECORD FROM THE SLIDE DATA FILE *)
(*                                       *)
(*****************************************)
VAR I:INTEGER;
BEGIN
  WITH SLIDE^ DO BEGIN
    TRLONG(DUM[1],SNUM);
    SEEK(SLIDE,TRUNC(SNUM-SCONST));   (* Find and read the record. *)
    GET(SLIDE);
    IF IOCHECK('SLIDE FILE') THEN BEGIN
      STR(CNUM,DUM[2]);
      STR(ANUM,DUM[3]);
      DUM[9]:=ACOMF;          (* Transfer the content of the record *)
      DUM[10]:=MORPH;         (* variables to the working variable. *)
      DUM[11]:=IDENT;
      DUM[12]:=MANY;
```

```
      DUM[13]:=SCOM;
      FOR I:=1 TO 2 DO BEGIN     (* Print the record on the screen. *)
        GOTOXY(13,LINES[I]);CRT(ERASEOL);
        WRITE(DUM[I]);
      END;
      FOR I:=9 TO 13 DO BEGIN
        GOTOXY(13,LINES[I]);CRT(ERASEOL);
        WRITE(DUM[I]);
      END;
      CRT(ERASEOS);
    END;
  END;
END;  (* GETSLIDE *)


PROCEDURE FINDAPHID;
(*********************************************)
(*                                           *)
(* SEARCH A NAME IN THE HOST-PLANT NAME FILE *)
(* AND CORRECT OR UPDATE THE BINARY TREE     *)
(*                                           *)
(*********************************************)

(* Identical to the procedure 'FINDHOST' except for the names of the *)
(* variables.  See the program 'HOST' for comments.                  *)

VAR PROX,RPROX,P:INTEGER;
    NPROX:STRING;
    ZZ:BOOLEAN;
    REC:INTEGER;
    PL,DU:STRING;


PROCEDURE SCRATCH;
(*************************************)
(*                                   *)
(* REMOVE A NODE FROM THE BINARY TREE *)
(*                                   *)
(*************************************)
BEGIN
  WITH APHID^ DO BEGIN
    IF LN<>0 THEN BEGIN
      IF HN<>0 THEN BEGIN
        SEEK(APHID,HN);
        GET(APHID);
        IF ALNOD<>0 THEN BEGIN
          REPEAT
            SEEK(APHID,ALNOD);
            GET(APHID);
          UNTIL ALNOD=0;
        END;
        ALNOD:=LN;
```

```
        SEEK(APHID,ANUM);
        PUT(APHID);
        SEEK(APHID,LN);
        LN:=ANUM;
        GET(APHID);
        ABNOD:=LN;
        SEEK(APHID,ANUM);
        PUT(APHID);
      END
      ELSE HN:=LN;
    END;
    IF BN<>0 THEN BEGIN
      SEEK(APHID,BN);
      GET(APHID);
      IF SIM THEN ASYN:=SYN
      ELSE IF ALNOD<>TRANSF(DUM[3]) THEN
              AHNOD:=HN
            ELSE ALNOD:=HN;
      SEEK(APHID,BN);
      PUT(APHID);
      END
    ELSE BEGIN
      BASE:=HN;
      SEEK(APHID,0);
      GET(APHID);
      APHID^.ABNOD:=BASE;
      SEEK(APHID,0);
      PUT(APHID);
    END;
    IF HN<>0 THEN BEGIN
      SEEK(APHID,HN);
      GET(APHID);
      ABNOD:=BN;
      SEEK(APHID,HN);
      PUT(APHID);
    END;
    IF SYN<>0 THEN BEGIN
      SEEK(APHID,SYN);
      GET(APHID);
      ABNOD:=BN;
      SEEK(APHID,SYN);
      PUT(APHID);
    END;
  END;
END;  (* SCRATCH *)
```

```
PROCEDURE SYNON;
(*******************)
(*                 *)
(* CREATE A SYNONYM *)
(*                 *)
(*******************)
VAR I:INTEGER;
BEGIN
  WITH APHID^ DO BEGIN
    GOTOXY(0,22);CRT(ERASEOS);
    WRITE(CHR(15),AGENUS,' ',ASPEC,' ',ASBSP,' in the aphid',
          ' file as #',REC,CHR(14));
    IF (SEL IN ['F','f']) AND ((BN<>0) OR (LN<>0) OR (HN<>0)
                OR (SIM=TRUE)) THEN BEGIN
      GOTOXY(0,21);
      WRITE(CHR(15),'Do you want to:create a S)ynonym, E)rase the record,');
      WRITE(' or K)eep it unchanged ?',CHR(14),' ');
      WW:=GETCHAR(['S','E','K','s','e','k']);
      CASE WW OF
          'E','e':BEGIN
                    SCRATCH;
                    SEEK(APHID,TRANSF(DUM[3]));
                    GET(APHID);
                    ASBGN:='';AGENUS:='';
                    ASPEC:='';ASBSP:='';ADESC:='';
                    ABNOD:=0;ALNOD:=0;AHNOD:=0;
                    ASYN:=0;ASIM:=FALSE;
                    SEEK(APHID,ANUM);
                    PUT(APHID);
                  END;
          'S','s':BEGIN
                    SCRATCH;
                    SEEK(APHID,REC);
                    GET(APHID);
                    SYN:=0;
                    IF ASYN<>0 THEN BEGIN
                      REPEAT
                        SEEK(APHID,ASYN);
                        GET(APHID);
                        REC:=ANUM;
                      UNTIL ((ASYN=0) OR (CONCAT(AGENUS,ASPEC,ASBSP)<DU));
                      IF (CONCAT(AGENUS,ASPEC,ASBSP))<DU THEN BEGIN
                        REC:=ABNOD;
                        ABNOD:=TRANSF(DUM[3]);
                        SYN:=ANUM;
                        SEEK(APHID,ANUM);
                        PUT(APHID);
                      END;
                    END;
                    SEEK(APHID,REC);
                    GET(APHID);
                    BN:=ANUM;
```

```
                    ASYN:=TRANSF(DUM[3]);
                    SEEK(APHID,REC);
                    PUT(APHID);
                    SEEK(APHID,TRANSF(DUM[3]));
                    GET(APHID);
                    ASIM:=TRUE;
                    ASYN:=SYN;
                    ABNOD:=BN;
                    ALNOD:=0;
                    AHNOD:=0;
                    SEEK(APHID,ANUM);
                    PUT(APHID);
                  END;
      END;
    END
    ELSE BEGIN
      STR(ANUM,DUM[3]);
      GOTOXY(13,LINES[3]);CRT(ERASEOL);
      WRITE(DUM[3]);
    END;
  NEWREC:=FALSE;
  BRANCH:=' ';
  EXIT(FINDAPHID);
  END;
END;  (* SYNON *)


BEGIN  (* FINDAPHID *)
  IF LASTAP<1 THEN EXIT (FINDAPHID);
  ZZ:=FALSE;
  WITH APHID^ DO BEGIN
    IF NOT (SEL IN['F','f']) THEN BEGIN
      SIM:=FALSE;
      SYN:=0;LN:=0;HN:=0;BN:=0;
    END;
    REC:=BASE;PROX:=70;RPROX:=0;NPROX:='';
    REPEAT
      SEEK(APHID,REC);
      GET(APHID);
      IF IOCHECK('APHID FILE') THEN BEGIN
        PL:=CONCAT(AGENUS,ASPEC,ASBSP);
        DU:=CONCAT(DUM[4],DUM[6],DUM[7]);
        P:=SIMIL(AGENUS,DUM[4])+SIMIL(ASPEC,DUM[6])+SIMIL(ASBSP,DUM[7]);
        IF P<=PROX THEN BEGIN
          PROX:=P;
          RPROX:=REC;
          NPROX:=CONCAT(AGENUS,' ',ASPEC,' ',ASBSP);
        END;
        IF PL=DU THEN SYNON
        ELSE BEGIN
          IF DU<PL THEN BEGIN
            IF ALNOD>0 THEN REC:=ALNOD
```

```
        ELSE BEGIN
          BRANCH:='L';
          ZZ:=TRUE;
        END;
      END
      ELSE BEGIN
        IF AHNOD>0 THEN REC:=AHNOD
        ELSE BEGIN
          BRANCH:='H';
          ZZ:=TRUE;
        END;
      END;
      IF ZZ THEN BEGIN
        GOTOXY(13,LINES[3]);CRT(ERASEOL);
        WRITE(DUM[3]);
        GOTOXY(25,LINES[3]);
        WRITE(CHR(15),'New aphid name',CHR(14));
        GOTOXY(0,22);CRT(ERASEOS);
        WRITE(CHR(15),'Closest aphid: ',RPROX,' ',NPROX,CHR(14));
        IF NOT (SEL IN ['f','F']) THEN BEGIN
          BN:=REC;
          EXIT(FINDAPHID);
        END
        ELSE BEGIN
          IF REC=TRANSF(DUM[3]) THEN EXIT(FINDAPHID);
          GOTOXY(0,21);
          CRT(ERASEOL);
          WRITE(CHR(15),'Save the changes ?',CHR(14),' ');
          IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
            WITH APHID^ DO BEGIN
              IF (BN<>0) OR (LN<>0) OR (HN<>0) OR (SYN<>0) THEN SCRATCH;
              BN:=REC;
              LN:=0;
              HN:=0;
              SYN:=0;
              SIM:=FALSE;
              STOREAP;
              SEEK(APHID,REC);
              GET(APHID);
              IF BRANCH='L' THEN ALNOD:=TRANSF(DUM[3])
              ELSE AHNOD:=TRANSF(DUM[3]);
              SEEK(APHID,REC);
              PUT(APHID);
            END;
          END;
          NEWREC:=FALSE;
          EXIT(FINDAPHID);
        END;
      END;
    END;
  END;
UNTIL FALSE;
```

```
  END;
END;  (* FINDAPHID *)


PROCEDURE CAPITAL(K:INTEGER);
(*************************************************)
(*                                               *)
(* CHANGE THE FIRST CHARACTER OF A STRING FOR THE *)
(* CORRESPONDING UPPER CASE CHARACTER AND THE     *)
(* OTHER CHARACTERS FOR THE CORRESPONDING LOWER   *)
(* CASE CHARACTERS.                               *)
(*                                               *)
(*************************************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE CORAPHID;
(*******************************)
(*                             *)
(* CORRECT THE APHID NAME FILE *)
(*                             *)
(*******************************)

(* Identical to the procedure 'CORPLANT' except for the names and  *)
(* indexes of the variables.  See the program 'HOST' for comments. *)

VAR YES:BOOLEAN;
    I:INTEGER;
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(25,0);
  WRITE(CHR(15),'CORRECTING THE APHID NAME FILE',CHR(14));
  REPEAT
  NEWREC:=TRUE;
    REPEAT
      GETSTRING(3,3);
    UNTIL (TRANSF(DUM[3])>0) AND (LASTAP)=TRANSF(DUM[3]));
    GETAPHID;
    IF IORES=0 THEN BEGIN
      GETSTRING(4,8);
      CAPITAL(4);
      CAPITAL(5);
      IF (APHID^.ABENUS<>DUM[4]) OR (APHID^.ASPEC<>DUM[6]) OR
         (APHID^.ASBSP<>DUM[7]) THEN FINDAPHID
      ELSE IF APHID^.ASIM THEN BEGIN
        GOTOXY(0,21);CRT(ERASEOL);
        WRITE(CHR(15),'Keep it as a synonym ?',CHR(14),' ');
        IF GETCHAR(['Y','N','y','n']) IN ['N','n'] THEN FINDAPHID;
      END;
      IF NEWREC THEN BEGIN
        GOTOXY(0,21);CRT(ERASEOL);
```

```
        WRITE(CHR(15),'Save the changes ?',CHR(14),' ');
        IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN STOREAP;
      END;
      GOTOXY(0,21);CRT(ERASEOS);
      WRITE(CHR(15),'Another record ??',CHR(14),' ');
      YES:=GETCHAR(['Y','y','N','n']) in ['n','N'];
    END
    ELSE YES:=TRUE;
  UNTIL YES;
END;  (* CORAPHID *)


PROCEDURE UPDAPHID;
(*****************************)
(*                           *)
(* UPDATE THE APHID NAME FILE *)
(*                           *)
(*****************************)

(* Identical to the procedure 'UPDPLANT' except for the names and  *)
(* indexes of the variables.  See the program 'HOST' for comments. *)

VAR YES:BOOLEAN;
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(26,0);
  WRITE(CHR(15),'UPDATING THE APHID NAME FILE',CHR(14));
  REPEAT
    IF LASTAP+1=MAXINT THEN BEGIN
      GOTOXY(0,21);CRT(ERASEOS);
      WRITE(CHR(15),'Largest acceptable aphid number reached.');
      WRITE(' Open a new file.',CHR(14));
      GOTOXY(0,22);
      WRITE(CHR(15),'Press spacebar to continue.',CHR(14));
      WW:=GETCHAR([' ']);
      EXIT(UPDAPHID);
    END;
    NEWREC:=TRUE;
    STR(LASTAP+1,DUM[3]);
    GOTOXY(13,LINES[3]);CRT(ERASEOL);
    WRITE(DUM[3]);
    GETSTRING(4,8);
    CAPITAL(4);
    CAPITAL(5);
    FINDAPHID;
    IF NEWREC THEN BEGIN
      GOTOXY(0,21);CRT(ERASEOL);
      WRITE(CHR(15),'Save the record ??',CHR(14),' ');
      IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN STOREAP;
    END;
    IF IORES=0 THEN BEGIN
      GOTOXY(0,21);CRT(ERASEOS);
```

```
      WRITE(CHR(15),'Another record ??',CHR(14),' ');
      YES:=GETCHAR(['Y','N','y','n']) IN ['N','n'];
    END
    ELSE YES:=TRUE;
  UNTIL YES;
END;  (* UPDAPHID *)


PROCEDURE CORSLIDE;
(*******************************)
(*                             *)
(* CORRECT THE SLIDE DATA FILE *)
(*                             *)
(*******************************)
VAR OK:BOOLEAN;
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(25,0);              (* Print the title *)
  WRITE(CHR(15),'CORRECTING THE SLIDE DATA FILE',CHR(14));
  REPEAT
    NEWREC:=FALSE;
    REPEAT
      OK:=FALSE;
      GETSTRING(1,1);        (* Read the slide reference number from *)
                             (* the keyboard.                        *)
      TRLONG(DUM[1],SLIDE^.SNUM);
      IF SLIDE^.SNUM>SCONST THEN
        IF SLIDE^.SNUM-SCONST<32767 THEN
          IF TRUNC(SLIDE^.SNUM-SCONST)<=LASTSL THEN
            OK:=TRUE;
    UNTIL OK=TRUE;  (* Repeat if the slide reference number is out   *)
                    (* of range.                                     *)
    GETSLIDE;       (* Read the slide data record from the file.     *)
    IF IORES<>0 THEN BYEBYE;
    GETAPHID;       (* If the I/O operation was successful, read the *)
                    (* aphid name record.                            *)
    IF IORES<>0 THEN BYEBYE;
    GETSTRING(2,13);(* Read the corrections from the keyboard.       *)
    CAPITAL(5);
    CAPITAL(4);
    IF (APHID^.AGENUS<>DUM[4]) OR (APHID^.ASPEC<>DUM[6])
       OR (APHID^.ASBSP<>DUM[7]) THEN BEGIN
      NEWREC:=TRUE; (* If the aphid name was changed, search for the *)
      FINDAPHID;    (* new name in the aphid name file.              *)
    END
    ELSE (* If the aphid genus, species or subspecies were not changed. *)
           IF (SLIDE^.ANUM<>TRANSF(DUM[3])) THEN
             IF (TRANSF(DUM[3])>0) AND (* If the aphid reference number*)
                                       (* was changed and it is within *)
                                       (* range, read the aphid record.*)
                       (TRANSF(DUM[3])<=LASTAP) THEN GETAPHID
             ELSE BEGIN  (* If out of range use the last reference number*)
```

```
            STR(APHID^.ANUM,DUM[3]);
            GOTOXY(13,LINES[3]);CRT(ERASEOL);
            WRITE(DUM[3]);
            WRITE(' ',CHR(15),'Out of range, ',
                  'previous number restored.',CHR(14));
          END
        ELSE BEGIN  (* Display an error message if the aphid subgenus *)
                    (* or describer was changed.                      *)
          IF APHID^.ASBGN<>DUM[5] THEN BEGIN
            DUM[5]:=APHID^.ASBGN;
            GOTOXY(13,LINES[5]);CRT(ERASEOL);
            WRITE(DUM[5]);
            WRITE('     ',CHR(15),'To correct use selection ',
                  'F in the main menu.',CHR(14));
          END;
          IF APHID^.ADESC<>DUM[8] THEN BEGIN
            DUM[8]:=APHID^.ADESC;
            GOTOXY(13,LINES[8]);CRT(ERASEOL);
            WRITE(DUM[8]);
            WRITE('     ',CHR(15),'To correct use selection ',
                  'F in the main menu.',CHR(14));
          END;
    END;
    GOTOXY(0,21);CRT(ERASEOL);
    WRITE(CHR(15),'Save the changes??',CHR(14),' ');
    IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
      IF NEWREC THEN BEGIN   (* Update the aphid name file if the  *)
        STR(LASTAP+1,DUM[3]);(* aphid name was changed for one not *)
        GOTOXY(13,LINES[3]); (* yet in the file.                   *)
        WRITE(DUM[3]);
        CRT(ERASEOL);
        STOREAP;
      END;
      IF IORES<>0 THEN BYEBYE
      ELSE STORESL;     (* Save the slide data record if *)
                        (* the aphid name record was     *)
                        (* properly saved.               *)
    END;
    GOTOXY(0,21);CRT(ERASEOS);
    WRITE(CHR(15),'Another record??',CHR(14),' ');
  UNTIL GETCHAR(['Y','N','y','n']) IN ['N','n'];
  BYEBYE;
END;  (* CORSLIDE *)
```

```
PROCEDURE UPDSLIDE;
(*****************************)
(*                           *)
(* UPDATE THE SLIDE DATA FILE *)
(*                           *)
(*****************************)
BEGIN
  GOTOXY(0,0);CRT(ERASEOL);
  GOTOXY(25,0);  (* Print the title *)
  WRITE(CHR(15),'UPDATING THE SLIDE DATA FILE',CHR(14));
  REPEAT
    IF ((LASTSL+1)=MAXINT) OR (SCONST+LASTSL>999998) THEN BEGIN
      GOTOXY(0,21);CRT(ERASEOS);  (* Exit if the largest slide reference *)
      IF LASTSL+1=MAXINT THEN     (* number is 32766.                    *)
          WRITE(CHR(15),'Slide data file is full.  Open a new file.',CHR(14))
      ELSE WRITE(CHR(15),'Largest slide number reached.',CHR(14));
      GOTOXY(0,22);
      WRITE('Press spacebar to continue.',CHR(14));
      WW:=GETCHAR([' ']);
      BYEBYE;
    END;
    NEWREC:=FALSE;
    IF LASTSL=0 THEN BEGIN  (* If working on the first record of a *)
      REPEAT                (* new file, read the reference number.*)
        GETSTRING(1,1);
        TRLONG(DUM[1],SLIDE^.SNUM);
      UNTIL SLIDE^.SNUM>0;
    END
    ELSE BEGIN                      (* If not the first record of a new  *)
    STR(LASTSL+SCONST+1,DUM[1]);  (* file print the slide reference    *)
    GOTOXY(13,LINES[1]);CRT(ERASEOL); (* number.                          *)
    WRITE(DUM[1]);
    END;
    GETSTRING(2,13);  (* Read the slide data record from the keyboard. *)
    CAPITAL(4);
    CAPITAL(5);
    IF (TRANSF(DUM[3])>0) AND   (* Read the aphid record from the file *)
                                (* if the aphid reference number is    *)
                                (* present and within range.           *)
                  (TRANSF(DUM[3])<=LASTAP) THEN GETAPHID
    ELSE BEGIN                  (* Otherwise, search in the aphid name *)
      NEWREC:=TRUE;             (* file for the aphid name.            *)
      FINDAPHID;
    END;
    GOTOXY(0,21);CRT(ERASEOL);
    WRITE(CHR(15),'Save the record??',CHR(14),' ');
    IF GETCHAR(['Y','N','y','n']) IN ['Y','y'] THEN BEGIN
      IF NEWREC THEN BEGIN     (* Update the aphid name file if the *)
        STR(LASTAP+1,DUM[3]);  (* aphid name is a new one.          *)
        GOTOXY(13,LINES[3]);
        WRITE(DUM[3]);
        CRT(ERASEOL);
```

```
        STOREAP;
      END;
      IF IORES=0 THEN STORESL  (* Update the slide data file.       *)
      ELSE BYEBYE;
    END;
    GOTOXY(0,21);CRT(ERASEOS);
    WRITE(CHR(15),'Another record??',CHR(14),' ');
  UNTIL GETCHAR(['Y','N','y','n']) IN ['N','n'];
  BYEBYE;
END;  (* UPDSLIDE *)


BEGIN (* SLIDEAPHID *)
  GETCVAL(DUM[1]);  (* Get the selection code from 'TITLE'. *)
  SEL:=DUM[1][1];
  GETCRTINFO;
  SETCHAIN('COLLECT:TITLE');  (* Select the next program to be executed*)
  RESET(FNAMES,'COLLECT:FILEINFO');
  INIT;            (* Initialization steps. *)
  CASE SEL OF
  'E','e':BEGIN       (* Update the aphid name file. *)
            DISP_E;
            UPDAPHID;
          END;
  'F','f':BEGIN       (* Correct the aphid name file. *)
            DISP_E;
            GOTOXY(0,21);  (* Print the largest aphid reference number. *)
            WRITE(CHR(15),'Largest aphid number is ',LASTAP,CHR(14));
            CORAPHID;
          END;
  'C','c':BEGIN       (* Update the slide data file. *)
            DISP_D;
            DISP_E;
            UPDSLIDE;
          END;
  'D','d':BEGIN       (* Correct the slide data file. *)
            DISP_D;
            DISP_E;
            GOTOXY(0,21);  (* Print the smallest and the largest     *)
                           (* slide reference number in the file.    *)
            WRITE(CHR(15),'Slide numbers range from ',SCONST+1,
                    ' to ',SCONST+LASTSL,CHR(14));
            CORSLIDE;
          END;
  END;
  BYEBYE
END.  (* SLIDEAPHID *)
```

```
PROGRAM HARDCOPY;
(****************)
(*              *)
(* PRINT A FILE *)
(*              *)
(****************)
USES CHAINSTUFF;
TYPE SETOFCHAR=SET OF CHAR;
    CRTCOMMAND= (ERASEOS,ERASEOL,UP,DOWN,RIGHT,LEFT,LEADIN);
    STLONG=STRING[255];
    LONG=INTEGER[6];
VAR COLL:FILE OF RECORD
            CNUM:INTEGER;
            DATE:STRING[12];
            CITY:   STRING[15];
            STATE:  STRING[15];
            COUNTRY: STRING[15];   (* See Table 4 for a description  *)
            COLLECT: STRING[20];   (* of the variables.              *)
            PNUM:INTEGER;
            PCOMF:STRING[1];
            LOCAT:STRING[50];
            COLONY:STRING[50];
            COLOUR:STRING[226];
            CCOM:STRING[226];
          END;
  PLANT:FILE OF PACKED RECORD
          PNUM:INTEGER;
          PFAM:STRING[20];
          PGENUS:STRING[20];
          PSPEC:STRING[20];     (* See Table 3 for a description *)
          PVAR:STRING[20];      (* of the variables.             *)
          PDESC:STRING[20];
          PSIM:BOOLEAN;
          PSYN:INTEGER;
          PHNOD:INTEGER;
          PLNOD:INTEGER;
          PBNOD:INTEGER;
        END;
   SLIDE:FILE OF RECORD
           SNUM:INTEGER[6];
           CNUM:INTEGER;
           ANUM:INTEGER;
           ACOMF:  STRING[1];  (* See Table 5 for a description *)
           MORPH: STRING[12];  (* of the variables.             *)
           IDENT: STRING[20];
           MANY:STRING[2];
           SCOM:STRING[65];
         END;
  APHID:FILE OF PACKED RECORD
          ANUM:INTEGER;
          AGENUS:STRING[20];
          ASBGN:STRING[20];
```

```
        ASPEC:STRING[20];
        ASBSP:STRING[20];   (* See Table 2 for a description *)
        ADESC:STRING[20];   (* of the variables.            *)
        ASIM:BOOLEAN;
        ASYN:INTEGER;
        AHNOD:INTEGER;
        ALNOD:INTEGER;
        ABNOD:INTEGER;
      END;
   FNAMES:FILE OF RECORD
          CFILE,PFILE,SFILE,AFILE:STRING;
        END;
   LINES:ARRAY[1..24] OF STRING[160];  (* Working variable containing *)
                                       (* the text to be printed.     *)
   TEMPO:STRING;
   ALPHA,              (* Aphid or host-plant names are printed *)
                       (* alphabetically, if true.              *)
   LSYN,               (* Print only the synonyms.            *)
   PNAME,              (* Print the host-plant name record.   *)
   ANAME,              (* Print the aphid name record.        *)
   CNAME:BOOLEAN;      (* Print the collection data record.   *)
   Y,                  (* Number of lines to be printed.      *)
   PAGE,               (* Number of lines printed on the page.*)
   LASTCOL,CCONST,     (* See the program 'HOST'.             *)
   I,                  (* Counter.                            *)
   FIRST,LAST,         (* First and last reference number to  *)
                       (* be printed.                         *)
   IORES:INTEGER;      (* I/O error code.                     *)
   SEL,                (* Main menu selection code.           *)
   WW:CHAR;            (* Dummy.                              *)
   BON,BOF,            (* Turn bold face character printing   *)
                       (* on and off respectively.            *)
   UON,UOF:STRING[6];(* Turn underlining on and off.          *)
   PRINTER:TEXT;
   CRTINFO: PACKED ARRAY[CRTCOMMAND] OF CHAR;
   PREFIXED:ARRAY[CRTCOMMAND] OF BOOLEAN;
(*$I-*)

PROCEDURE GETCRTINFO;
(*****************************************)
(*                                       *)
(* READ THE SCREEN CONTROL CHARACTERS FROM *)
(* THE SYSTEM.MISCINFO FILE              *)
(*                                       *)
(*****************************************)

(* See the program 'HOST' for the listing and comments. *)
```

```
PROCEDURE CRT(C:CRTCOMMAND);
(*************************)
(*                       *)
(*    SCREEN COMMANDS    *)
(*                       *)
(*************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE BYEBYE;
(*****************************************)
(*                                       *)
(* UPDATE FILEINFO AND CLOSE THE FILES *)
(*                                       *)
(*****************************************)
BEGIN
  SEEK(FNAMES,0);
  PUT(FNAMES);
  CLOSE(APHID,LOCK);
  CLOSE(PLANT,LOCK);
  CLOSE(COLL,LOCK);
  CLOSE(SLIDE,LOCK);
  CLOSE(FNAMES,LOCK);
  EXIT(PROGRAM);
END;  (* BYEBYE *)


PROCEDURE TRLONG(VAR WW:STRING;VAR TEMP:LONG);
(*****************************************************)
(*                                                   *)
(* CONVERT THE VALUE OF A STRING INTO A LONG INTEGER *)
(*                                                   *)
(*****************************************************)

(* See the program 'SLIDEAPHID' for the listing and comments. *)


FUNCTION TRANSF(Z:STRING):INTEGER;
(*************************************************)
(*                                               *)
(* CONVERT THE VALUE OF A STRING INTO AN INTEGER *)
(*                                               *)
(*************************************************)

(* See the program 'HOST' for the listing and comments. *)
```

```
FUNCTION GETCHAR(OK:SETOFCHAR):CHAR;
(*************************************)
(*                                   *)
(* GET A CHARACTER FROM THE KEYBOARD *)
(*                                   *)
(*************************************)

(* See the program 'TITLE' for the listing and comments. *)


FUNCTION IOCHECK(F:STRING):BOOLEAN;
(*************************************)
(*                                   *)
(* VERIFY THE STATUS OF THE LAST I/O *)
(* OPERATION AND PRINT AN ERROR      *)
(* MESSAGE IF THE OPERATION WAS NOT  *)
(* SUCCESSFUL                        *)
(*                                   *)
(*************************************)

(* See the program 'HOST' for the listing and comments. *)


FUNCTION YES:BOOLEAN;
(*************************************************)
(*                                               *)
(* TRUE IF THE CHARACTER ENTERED IS 'Y' OR 'y' *)
(*                                               *)
(*************************************************)
BEGIN
  YES:=GETCHAR(['Y','N','y','n']) IN ['Y','y'];
END;  (* YES *)


PROCEDURE SELECT;
(*************************************)
(*                                   *)
(* SELECTION OF THE FILE TO BE PRINTED *)
(*                                   *)
(*************************************)
VAR LASTPL,LASTAP,LASTSL:INTEGER;
    SCONST:INTEGER[6];


PROCEDURE OPEN (A:CHAR;L:INTEGER);
(*********************************)
(*                               *)
(*     NAME AND OPEN FILES       *)
(*                               *)
(*********************************)

(* See programs 'HOST' and 'SLIDEAPHID' for comments. *)
```

```
VAR RES:BOOLEAN;
    CHOI:CHAR;
    X:INTEGER;


PROCEDURE OP1;
(*******************************)
(*                             *)
(*     GET THE FULL FILE NAME  *)
(*                             *)
(*******************************)
VAR RENAME:BOOLEAN;
BEGIN
  GOTOXY(0,L);CRT(ERASEOL);
  CASE A OF
    'A':BEGIN
          WRITE(CHR(15),'Collection data file=',CHR(14));
          IF (FNAMES^.CFILE='')  THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.CFILE,'  ');
            RENAME:=NOT (YES);
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(22,L);CRT(ERASEOL);
            READLN(FNAMES^.CFILE);
          END;
        END;
    'C':BEGIN
          WRITE(CHR(15),'Host-plant name file=',CHR(14));
          IF (FNAMES^.PFILE='')  THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.PFILE,'  ');
            RENAME:=NOT (YES);
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(22,L);CRT(ERASEOL);
            READLN(FNAMES^.PFILE);
          END;
        END;
    'B':BEGIN
          WRITE(CHR(15),'Slide data file=',CHR(14));
          IF (FNAMES^.SFILE='')  THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.SFILE,'  ');
            RENAME:=NOT (YES);
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(17,L);CRT(ERASEOL);
            READLN(FNAMES^.SFILE);
          END;
        END;
```

```
    'D':BEGIN
          WRITE(CHR(15),'Aphid name file=',CHR(14));
          IF (FNAMES^.AFILE='')  THEN RENAME:=TRUE
          ELSE BEGIN
            WRITE(' ',FNAMES^.AFILE,'  ');
            RENAME:=NOT (YES);
          END;
          IF RENAME=TRUE THEN BEGIN
            GOTOXY(17,L);CRT(ERASEOL);
            READLN(FNAMES^.AFILE);
          END;
        END;
  END;
END;  (* OP1 *)


BEGIN  (* OPEN *)
  RES:=FALSE;
  REPEAT
    OP1;
    CASE A OF
      'A':RESET(COLL,FNAMES^.CFILE);
      'B':RESET(SLIDE,FNAMES^.SFILE);
      'C':RESET(PLANT,FNAMES^.PFILE);
      'D':RESET(APHID,FNAMES^.AFILE);
    END;
    RES:=IOCHECK(' ');
    IF NOT (IORES IN [5,6,9]) THEN BEGIN
      IF RES=FALSE THEN BEGIN
        GOTOXY(0,L+2);CRT(ERASEOL);
        WRITE(CHR(15),'Choose between: C)hange the file name, ',
             'R)eturn to main menu.',CHR(14),' ');
        CHOI:=GETCHAR(['C','R','c','r']);
        CASE CHOI OF
          'R','r':BYEBYE;
        END;
      END
      ELSE BEGIN
        CASE A OF
          'A':BEGIN
                IF COLL^.CITY <>'C' THEN BEGIN
                  GOTOXY(0,L+2);CRT(ERASEOL);
                  WRITE(CHR(15),FNAMES^.CFILE,' is not',
                        ' a collection data file.',CHR(14));
                  RES:=FALSE;
                  CLOSE(COLL,LOCK);
                END
                ELSE BEGIN
                  LASTCOL:=COLL^.CNUM;
                  CCONST:=COLL^.PNUM;
                END;
              END;
```

```
        'B':BEGIN
              IF SLIDE^.ACOMF <>'S' THEN BEGIN
                GOTOXY(0,L+2);CRT(ERASEOL);
                WRITE(CHR(15),FNAMES^.SFILE,' is not',
                      ' a slide data file.',CHR(14));
                RES:=FALSE;
                CLOSE(SLIDE,LOCK);
              END
              ELSE BEGIN
                LASTSL:=SLIDE^.CNUM;
                SCONST:=SLIDE^.SNUM;
              END;
            END;
        'C':BEGIN
              IF PLANT^.PFAM <>'P' THEN BEGIN
                GOTOXY(0,L+2);CRT(ERASEOL);
                WRITE(CHR(15),FNAMES^.PFILE,' is not',
                      ' a host-plant name file.',CHR(14));
                RES:=FALSE;
                CLOSE(PLANT,LOCK);
              END
              ELSE LASTPL:=PLANT^.PNUM;
            END;
        'D':BEGIN
              IF APHID^.AGENUS<>'A' THEN BEGIN
                GOTOXY(0,L+2);CRT(ERASEOL);
                WRITE(CHR(15),FNAMES^.AFILE,' is not',
                      ' an aphid name file.',CHR(14));
                RES:=FALSE;
                CLOSE(APHID,LOCK);
              END
              ELSE LASTAP:=APHID^.ANUM;
            END;
      END;
    END;
  END;
 UNTIL RES;
END;  (* OPEN *)


PROCEDURE INITCOLL;
(*************************************)
(*                                   *)
(* OPTIONS AVAILABLE FOR PRINTING THE  *)
(* COLLECTION DATA FILE                *)
(*                                   *)
(*************************************)
BEGIN
  GOTOXY(10,12);
  WRITE(CHR(15),'OPTIONS:',CHR(14));
  GOTOXY(15,14);
  WRITE ('Print all of the file:');
```

```
  GOTOXY(15,15);
  WRITE('First desired number:');
  GOTOXY(15,16);
  WRITE('Last desired number:');
  GOTOXY(15,17);
  WRITE('Print only the plant number?');
  GOTOXY(38,14);
  IF YES THEN BEGIN
    FIRST:=1;                  (* Definition of the first and the last *)
    LAST:=LASTCOL-CCONST;      (* collection reference numbers to be   *)
  END                          (* printed.  It can be the whole file   *)
  ELSE BEGIN                   (* or part of it.                       *)
    REPEAT
      REPEAT
        GOTOXY(37,15);CRT(ERASEOL);
        READLN(TEMPO);
        FIRST:=TRANSF(TEMPO);
      UNTIL (FIRST>CCONST) AND (FIRST<=(LASTCOL+CCONST));
      REPEAT
        GOTOXY(36,16);CRT(ERASEOL);
        READLN(TEMPO);
        LAST:=TRANSF(TEMPO);
      UNTIL (LAST>CCONST) AND (LAST<=(LASTCOL+CCONST));
    UNTIL FIRST<=LAST;
    FIRST:=FIRST-CCONST;
    LAST:=LAST-CCONST;
  END;
  GOTOXY(43,17);       (* Will print the host-plant reference number *)
  PNAME:=NOT (YES);    (* only or the whole record.                  *)
END;  (* INITCOLL *)


PROCEDURE INITHOST;
(*************************************************************)
(*                                                           *)
(* OPTIONS AVAILABLE FOR PRINTING THE HOST-PLANT NAME FILE   *)
(*                                                           *)
(*************************************************************)
BEGIN
  GOTOXY(10,12);
  WRITE(CHR(15),'OPTIONS:',CHR(14));
  GOTOXY(15,14);
  WRITE('Print only the synonyms:');
  GOTOXY(15,15);
  WRITE('Print the names alphabetically?');
  GOTOXY(15,16);
  WRITE('Print all of the file:');
  GOTOXY(15,17);
  WRITE('First desired number:');
  GOTOXY(15,18);
  WRITE('Last desired number:');
  GOTOXY(40,14);
```

```
  IF YES THEN BEGIN   (* Will print the list of synonyms only. *)
    LSYN:=TRUE;
    FIRST:=1;
    LAST:=LASTPL;
  END
  ELSE BEGIN          (* Will print the names alphabetically.  *)
    GOTOXY(47,15);
    IF YES THEN ALPHA:=TRUE
    ELSE BEGIN
      GOTOXY(38,16);    (* Definition of the first and the last *)
      IF YES THEN BEGIN (* reference numbers to be printed.     *)
        FIRST:=1;
        LAST:=LASTPL;
      END
      ELSE BEGIN
        REPEAT
          REPEAT
            GOTOXY(37,17);CRT(ERASEOL);
            READLN(TEMPO);
            FIRST:=TRANSF(TEMPO);
          UNTIL (FIRST>0) AND (FIRST<=LASTPL);
          REPEAT
            GOTOXY(37,18);CRT(ERASEOL);
            READLN(TEMPO);
            FIRST:=TRANSF(TEMPO);
          UNTIL (LAST>0) AND (LAST<=LASTPL);
        UNTIL FIRST<=LAST;
      END;
    END;
  END;
END;  (* INITHOST *)


PROCEDURE INITAP;
(*********************************)
(*                               *)
(* OPTIONS AVAILABLE FOR PRINTING *)
(* THE APHID NAME FILE            *)
(*                               *)
(*********************************)
BEGIN
  GOTOXY(10,12);
  WRITE(CHR(15),'OPTIONS:',CHR(14));
  GOTOXY(15,14);
  WRITE('Print only the synonyms:');
  GOTOXY(15,15);
  WRITE('Print the names alphabetically?');
  GOTOXY(15,16);
  WRITE('Print all of the file?');
  GOTOXY(15,17);
  WRITE('First desired number:');
  GOTOXY(15,18);
```

```
  WRITE('Last desired number:');
  GOTOXY(40,14);
  IF YES THEN BEGIN   (* Will print the synonyms list only.  *)
    LSYN:=TRUE;
    FIRST:=1;
    LAST:=LASTAP;
  END
  ELSE BEGIN
    GOTOXY(47,15);
    IF YES THEN ALPHA:=TRUE  (* Will print the names alphabetically *)
    ELSE BEGIN               (* Will print the whole file.          *)
      GOTOXY(38,16);
      IF YES THEN BEGIN
        FIRST:=1;
        LAST:=LASTAP;
      END
      ELSE BEGIN             (* Selection of the first and the last *)
        REPEAT               (* reference numbers to be printed.    *)
          REPEAT
            GOTOXY(37,17);CRT(ERASEOL);
            READLN(TEMPO);
            FIRST:=TRANSF(TEMPO);
          UNTIL (FIRST>0) AND (FIRST<=LASTAP);
          REPEAT
            GOTOXY(36,18);CRT(ERASEOL);
            READLN(TEMPO);
            FIRST:=TRANSF(TEMPO);
          UNTIL (LAST>0) AND (LAST<=LASTAP);
        UNTIL FIRST<=LAST;
      END;
    END;
  END;
END;  (* INITAP *)


PROCEDURE INITSL;
(*********************************)
(*                               *)
(* OPTIONS AVAILABLE FOR PRINTING *)
(* THE SLIDE DATA FILE            *)
(*                               *)
(*********************************)
VAR L1,L2:INTEGER[6];
    OK:BOOLEAN;
BEGIN
  GOTOXY(10,12);
  WRITE(CHR(15),'OPTIONS;',CHR(14));
  GOTOXY(15,14);
  WRITE('Print all of the file:');
  GOTOXY(15,15);
  WRITE('First desired number:');
  GOTOXY(15,16);
```

```
  WRITE('Last desired number:');
  GOTOXY(15,17);
  WRITE('Print only the aphid number:');
  GOTOXY(15,18);
  WRITE('Print only the collection number?');
  GOTOXY(38,14);
  IF YES THEN BEGIN    (* Will print the whole file. *)
    FIRST:=1;
    LAST:=LASTSL;
  END
  ELSE BEGIN           (* Selection of the first and the last *)
    REPEAT             (* reference numbers to be printed.    *)
      REPEAT
        OK:=FALSE;
        GOTOXY(37,15);CRT(ERASEOL);
        READLN(TEMPO);
        TRLONG(TEMPO,L1);
        IF L1>SCONST THEN
          IF (L1-SCONST)<32767 THEN
            IF (TRUNC(L1-SCONST))<=LASTSL THEN
              OK:=TRUE;
      UNTIL OK=TRUE;
      REPEAT
        OK:=FALSE;
        GOTOXY(36,16);CRT(ERASEOL);
        READLN(TEMPO);
        TRLONG(TEMPO,L2);
        IF L2>SCONST THEN
          IF (L2-SCONST)<32767 THEN
            IF (TRUNC(L2-SCONST))<=LASTSL THEN
              OK:=TRUE;
      UNTIL OK=TRUE;
    UNTIL L1<=L2;
    FIRST:=TRUNC(L1-SCONST);
    LAST:=TRUNC(L2-SCONST);
  END;
  GOTOXY(44,17);     (* Can print the entire aphid name record. *)
  ANAME:=NOT (YES);
  GOTOXY(49,18);
  IF NOT (YES)  THEN BEGIN  (* Can print the entire collection record. *)
    CNAME:=TRUE;
    GOTOXY(15,19);
    WRITE('Print only the host number?');
    PNAME:=NOT (YES);       (* Can print the entire host-plant record. *)
  END
  ELSE CNAME:=FALSE;
END;  (* INITSL *)


BEGIN    (*SELECT*)
  GOTOXY(10,2);
  WRITE(CHR(15),'Print which file??',CHR(14));
```

```
  GOTOXY(15,4);
  WRITE('A.- The collection data file.');
  GOTOXY(15,5);
  WRITE('B.- The slide data file.');
  GOTOXY(15,6);
  WRITE('C.- The host-plant name file.');
  GOTOXY(15,7);
  WRITE('D.- The aphid name file.');
  GOTOXY(15,8);
  WRITE('E.- Return to main menu.');
  GOTOXY(10,10);
  WRITE(CHR(15),'Enter your selection:',CHR(14),' ');
  SEL:=GETCHAR(['A'..'E','a'..'e']);
  CASE SEL OF
    'A','a':BEGIN   (* Print the collection data file. *)
              OPEN('A',20);
              INITCOLL;
              IF PNAME THEN OPEN('C',20);
            END;
    'B','b':BEGIN   (* Print the slide data file. *)
              OPEN('B',20);
              INITSL;
              IF CNAME THEN OPEN('A',20);
              IF ANAME THEN OPEN('D',20);
              IF PNAME THEN OPEN('C',20);
            END;
    'C','c':BEGIN   (* Print the host-plant name file. *)
              OPEN('C',20);
              INITHOST;
            END;
    'D','d':BEGIN   (* Print the aphid name file. *)
              OPEN('D',20);
              INITAP;
            END;
    'E','e':BYEBYE;
  END;
END;  (* SELECT *)


PROCEDURE LONGLINE(LN:STLONG);
(*****************************************************)
(*                                                   *)
(* SPLIT A STRING INTO STRINGS OF 72 CHARACTERS (max) *)
(*                                                   *)
(*****************************************************)
VAR E:INTEGER;
BEGIN
  REPEAT
    Y:=Y+1;
    E:=73;
    IF LENGTH(LN)>72 THEN BEGIN
      REPEAT       (* The source string will be split after the blank  *)
```

```
        E:=E-1;    (* the period or the comma closest to the 73th      *)
                   (* character.                                        *)
      UNTIL LN[E] IN [' ','.',',',];
      LINES[Y]:=COPY(LN,1,E);
      DELETE(LN,1,E);
      IF LENGTH(LN)>0 THEN LN:=CONCAT('      ',LN);
    END
    ELSE BEGIN
      LINES[Y]:=LN;
      EXIT(LONGLINE);
    END;
  UNTIL FALSE;
END;  (* LONGLINE *)


PROCEDURE SENDPRT;
(*************************************************)
(*                                               *)
(* SEND THE FORMATTED INFORMATION TO THE PRINTER *)
(*                                               *)
(*************************************************)
VAR I:INTEGER;


PROCEDURE HEXCODE(J:INTEGER);
(*************************************************)
(*                                               *)
(* FIND AND CONVERT HEXADECIMAL FORMATTING CODE  *)
(* TO ASCII                                      *)
(*                                               *)
(*************************************************)
CONST HEXFLAG='$';   (* Establish the flag character. *)
      HEX='123456789ABCDEF'; (* Define hexadecimal digits. *)
VAR PLACE,DIGIT,VALUE:INTEGER;
    TEM:STRING;
BEGIN
  TEM:='0';
  DIGIT:=0;
  FOR PLACE:=1 TO LENGTH(LINES[J]) DO BEGIN
  (* Send the string one character at a time to the printer.   *)
  (* If a flag character is found, convert the two characters  *)
  (* following in ASCII and send the value to the printer.     *)
    TEM[1]:=LINES[J][PLACE];
    IF TEM[1]=HEXFLAG THEN DIGIT:=2
    ELSE IF DIGIT=2 THEN BEGIN
      VALUE:=POS(TEM,HEX)*16;
      DIGIT:=DIGIT-1;
    END
    ELSE IF DIGIT=1 THEN BEGIN
          VALUE:=VALUE+POS(TEM,HEX);
          WRITE(PRINTER,CHR(VALUE));
          DIGIT:=DIGIT-1;
```

```
        END
        ELSE IF DIGIT=0 THEN WRITE(PRINTER,TEM[1]);
  END;
  WRITELN(PRINTER);
END;  (* HEXCODE *)


BEGIN  (* SENDPRT *)
  IF (PAGE+Y>60) AND (PAGE>54) THEN BEGIN
    WRITE(PRINTER,CHR(12));  (* Execute a form feed if the page        *)
    PAGE:=0;                 (* contains more than 54 lines and the    *)
  END;                       (* number of lines printed plus the number*)
                             (* to be printed is larger than 60.       *)
  FOR I:=1 TO Y DO BEGIN
    PAGE:=PAGE+1;
    IF PAGE>60 THEN BEGIN     (* Print 60 lines/page.                  *)
      WRITE(PRINTER,CHR(12));
      PAGE:=1;
    END;
    HEXCODE(I);
  END;
END;  (* SENDPRT *)


(* The next four 'PRT....' procedures format the four record types. *)
(* The processing consists (usually) in verifying if the variable   *)
(* is empty, verifying if the string is short enough to be added at *)
(* the end of the previously processed line and inserting printing  *)
(* commands (bold face, underlining etc.).                          *)

PROCEDURE PRTPLANT;
(************************************)
(*                                  *)
(* FORMAT THE HOST-PLANT NAME RECORD *)
(*                                  *)
(************************************)
VAR L,I:INTEGER;
    W:STRING[5];
    SYNOF:BOOLEAN;


PROCEDURE PRTSYN;
(************************************)
(*                                  *)
(* ADD BLANKS BEFORE THE CORRECT    *)
(* NAME OF A SYNONYM                *)
(*                                  *)
(************************************)
BEGIN
  IF (ALPHA=TRUE) AND (PLANT^.PSIM=TRUE) THEN BEGIN
    LINES[Y]:=CONCAT('     ',LINES[Y]);
    L:=L+5;
```

```
  END;
END;  (* PRTSYN *)


PROCEDURE PARTONE;
(*********************)
(*                   *)
(* PRTPLANT, PART ONE *)
(*                   *)
(*********************)
BEGIN
  WITH PLANT^ DO BEGIN
    Y:=Y+1;
    STR(PNUM,W);  (* Process the host-plant reference number. *)
    IF LENGTH(W)<5 THEN
      FOR I:=LENGTH(W) TO 4 DO W:=CONCAT(' ',W);
    L:=5;
    LINES[Y]:=CONCAT(BON,W,BOF);
    PRTSYN;
    IF SYNOF=TRUE THEN BEGIN  (* If printing the file by reference   *)
                              (* number and the present record is the*)
                              (* correct name of a synonym just      *)
                              (* processed.                          *)
      LINES[Y]:=CONCAT('  Syn. of:',LINES[Y]);
      L:=L+10;
    END;
    IF PGENUS<>'' THEN BEGIN  (* Process the host-plant genus. *)
      L:=L+LENGTH(PGENUS)+1;
      LINES[Y]:=CONCAT(LINES[Y],' ',UON,PGENUS,UOF);
    END
    ELSE BEGIN                (* If the host-plant genus is missing, *)
                              (* the record is considered empty.     *)
      LINES[Y]:=CONCAT(LINES[Y],' ','EMPTY');
      EXIT(PRTPLANT);
    END;
    IF PSPEC<>'' THEN BEGIN   (* Process the host-plant species.     *)
      L:=L+LENGTH(PSPEC)+1;
      LINES[Y]:=CONCAT(LINES[Y],' ',UON,PSPEC,UOF);
    END
    ELSE BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' ',UON,'sp.',UOF);
      L:=L+4;
    END;
    IF PVAR<>'' THEN BEGIN    (* Process the host-plant variety.     *)
      L:=L+6+LENGTH(PVAR);
      LINES[Y]:=CONCAT(LINES[Y],' var. ',UON,PVAR,UOF);
    END;
    IF SEL IN ['A','B','a','b'] THEN   (* If printing the host-plant *)
      IF PNAME=TRUE THEN               (* name as an option while    *)
        IF COLL^.PCOMF<>'' THEN BEGIN  (* printing the collection    *)
                                       (* data file or the slide data*)
                                       (* file.                      *)
```

```
          L:=L+2;
          LINES[Y]:=CONCAT(LINES[Y],' ',COLL^.PCOMF);
        END;
    IF PDESC<>'' THEN          (* Process the host-plant describer.   *)
      IF (L+LENGTH(PDESC)+1)>72 THEN BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT('            ',PDESC);
        L:=LENGTH(PDESC)+10;
        PRTSYN;
      END
      ELSE BEGIN
        LINES[Y]:=CONCAT(LINES[Y],' ',PDESC);
        L:=L+1+LENGTH(PDESC);
      END;
  END;
END;  (* PARTONE *)


PROCEDURE PARTTWO;
(**********************)
(*                    *)
(* PRTPLANT, PART TWO *)
(*                    *)
(**********************)
BEGIN
  WITH PLANT^ DO BEGIN
    IF PFAM<>'' THEN * (* Process the host-plant family. *)
      IF (L+3+LENGTH(PFAM))>72 THEN BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT('            (',PFAM,')');
        PRTSYN;
      END
      ELSE LINES[Y]:=CONCAT(LINES[Y],' (',PFAM,')');
  END;
END;  (* PARTTWO *)


BEGIN  (*PRTPLANT*)
  SYNOF:=FALSE;
  PARTONE;
  PARTTWO;
  IF (PLANT^.PSIM=TRUE) AND (ALPHA=FALSE) AND (LSYN=FALSE) THEN BEGIN
    (* If not printing the host-plant names alphabetically or just *)
    (* the synonyms list and the name is a synonym.                *)
    REPEAT
      SEEK(PLANT,PLANT^.PBNOD);  (* Find the correct name.         *)
      GET(PLANT);
    UNTIL PLANT^.PSIM=FALSE;
    SYNOF:=TRUE;
    PARTONE;                     (* Print the correct name.        *)
    PARTTWO;
  END;
```

```
END;  (* PRTPLANT *)


PROCEDURE PRTAPHID;
(*******************************)
(*                             *)
(* FORMAT THE APHID NAME RECORD *)
(*                             *)
(*******************************)
VAR L,              (* Number of digits of the line. *)
    I:INTEGER;
    W:STRING[5];
    SYNOF:BOOLEAN;(* True if the record is the correct name of   *)
                  (* the last aphid name processed.  Used only   *)
                  (* when printing the names by reference number *)
                  (* or as an option.                            *)


PROCEDURE PRTSYN;
(*******************************)
(*                             *)
(* ADD BLANKS BEFORE A SYNONYM *)
(*                             *)
(*******************************)
BEGIN
  (* Use only when printing the names alphabetically. *)
  IF (ALPHA=TRUE) AND (APHID^.ASIM=TRUE) THEN BEGIN
    LINES[Y]:=CONCAT('      ',LINES[Y]);
    L:=L+5;
  END;
END;  (* PRTSYN *)


PROCEDURE PARTA;
(*********************)
(*                   *)
(* PRTAPHID, PART ONE *)
(*                   *)
(*********************)
BEGIN
  WITH APHID^ DO BEGIN
    Y:=Y+1;
    STR(ANUM,W);        (* Process the aphid name reference number. *)
    IF LENGTH(W)<5 THEN
      FOR I:=LENGTH(W) TO 4 DO W:=CONCAT(' ',W);
    L:=5;
    LINES[Y]:=CONCAT(BON,W,BOF);
    PRTSYN;
    IF SYNOF=TRUE THEN BEGIN     (* Add a title line if processing  *)
                                 (* the correct name of a synonym   *)
                                 (* while printing the aphid names  *)
                                 (* by reference number or as an    *)
```

```
                                  (* option.                          *)
      LINES[Y]:=CONCAT('  Syn. of:',LINES[Y]);
      L:=L+10;
    END;
    IF AGENUS<>'' THEN BEGIN    (* Process the aphid genus.          *)
      L:=L+LENGTH(AGENUS)+1;
      LINES[Y]:=CONCAT(LINES[Y],' ',UON,AGENUS,UOF);
    END
    ELSE BEGIN    (* The record is considered empty if the aphid   *)
                  (* genus is missing.                              *)
      LINES[Y]:=CONCAT(LINES[Y],' ','EMPTY');
      EXIT(PRTAPHID);
    END;
    IF ASBGN<>'' THEN BEGIN    (* Process the aphid subgenus.      *)
      L:=L+3+LENGTH(ASBGN);
      LINES[Y]:=CONCAT(LINES[Y],' (',UON,ASBGN,UOF,')');
    END;
    IF ASPEC<>'' THEN BEGIN    (* Process the aphid species.       *)
      IF (L+1+LENGTH(ASPEC))<=72 THEN BEGIN
        L:=L+1+LENGTH(ASPEC);
        LINES[Y]:=CONCAT(LINES[Y],' ',UON,ASPEC,UOF);
      END
      ELSE BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT('           ',UON,ASPEC,UOF);
        L:=10+LENGTH(ASPEC);
        PRTSYN;
      END;
    END
    ELSE BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' ',UON,'sp.',UOF);
      L:=L+4;
    END;
    IF SEL IN ['b','B'] THEN  (* If printing the aphid name as an option*)
      IF SLIDE^.ACOMF<>'' THEN BEGIN
        LINES[Y]:=CONCAT(LINES[Y],' ',SLIDE^.ACOMF);
        L:=L+2;
      END;
  END;
END;  (* PARTA *)


PROCEDURE PARTB;
(**********************)
(*                    *)
(* PRTAPHID, PART TWO *)
(*                    *)
(**********************)
BEGIN
  WITH APHID^ DO BEGIN
    IF ASBSP<>'' THEN   (* Process the aphid subspecies. *)
      IF L+8+LENGTH(ASBSP)>72 THEN BEGIN
```

```
      Y:=Y+1;
      LINES[Y]:=CONCAT('            subsp. ',UON,ASBSP,UOF);
      L:=17+LENGTH(ASBSP);
      PRTSYN;
    END
    ELSE BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' subsp. ',UON,ASBSP,UOF);
      L:=L+8+LENGTH(ASBSP);
    END;
    IF ADESC<>'' THEN   (* Process the aphid describer. *)
      IF (L+1+LENGTH(ADESC))>72 THEN BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT('            ',ADESC);
        L:=10+LENGTH(ADESC);
        PRTSYN;
      END
      ELSE BEGIN
        LINES[Y]:=CONCAT(LINES[Y],' ',ADESC);
        L:=1+L+LENGTH(ADESC);
      END;
  END;
END;  (* PARTB *)


BEGIN  (*PRTAPHID*)
  SYNOF:=FALSE;
  PARTA;
  PARTB;
  IF (APHID^.ASIM=TRUE) AND (ALPHA=FALSE) AND (LSYN=FALSE) THEN BEGIN
    (* If processing a synonym while printing the file by reference *)
    (* number or as an option.                                      *)
    REPEAT
      SEEK(APHID,APHID^.ABNOD);  (* Find the correct name. *)
      GET(APHID);
    UNTIL APHID^.ASIM=FALSE;
    SYNOF:=TRUE;
    PARTA;                       (* Print the correct name. *)
    PARTB;
  END;
END;  (* PRTAPHID *)


PROCEDURE PRTCOLL;
(*********************************)
(*                               *)
(* FORMAT A COLLECTION DATA RECORD *)
(*                               *)
(*********************************)
VAR I:INTEGER;
    W:STRING[5];
    Z:STRING[12];
    V:STRING[255];
```

```
BEGIN
  WITH COLL^ DO BEGIN
    Y:=Y+1;          (* Process the collection reference number. *)
    STR(CNUM,W);
    IF LENGTH(W)<5 THEN
      FOR I:=LENGTH(W) TO 4 DO W:=CONCAT(W,' ');
    IF DATE='' THEN Z:='          '
    ELSE Z:=DATE;   (* Process the date and the collector. *)
    LINES[Y]:=CONCAT(BON,'COLL.#:',W,BOF,'     Date:',Z,'     ',COLLECT);
    Y:=Y+1;          (* Process the location variables. *)
    LINES[Y]:=CONCAT(CITY,'     ',STATE,'     ',COUNTRY);
    IF PNAME THEN PRTPLANT  (* If the option was selected, print *)
    ELSE BEGIN              (* the whole host-plant record.      *)
      Y:=Y+1;
      STR(PNUM,W);          (* Process the aphid name reference number*)
      LINES[Y]:=CONCAT('Plant #:',BON,W,BOF,' ',PCOMF);
    END;
    IF LOCAT<>'' THEN BEGIN (* Process the 'location on the host' data*)
      Y:=Y+1;
      LINES[Y]:=CONCAT('Location on the host: ',LOCAT);
    END;
    IF COLONY<>'' THEN BEGIN (* Process the colony size. *)
      Y:=Y+1;
      LINES[Y]:=CONCAT('Colony: ',COLONY);
    END;       (* Process the colour and comments. *)
    IF COLOUR<>'' THEN LONGLINE(CONCAT('Colour: ',COLOUR));
    IF CCOM<>'' THEN LONGLINE(CONCAT('Comments: ',CCOM));
    IF SEL IN ['A','a'] THEN BEGIN
      Y:=Y+1;         (* Add a blank line if the collection record  *)
      LINES[Y]:=' '; (* is not printed as an option.               *)
    END;
  END;
END;  (* PRTCOLL *)


PROCEDURE PRTSLIDE;
(*****************************)
(*                           *)
(* FORMAT A SLIDE DATA RECORD *)
(*                           *)
(*****************************)
VAR I:INTEGER;
    W:STRING[6];
BEGIN
  WITH SLIDE^ DO BEGIN
    Y:=Y+1;
    STR(SNUM,W);  (* Process the slide reference number and   *)
                  (* the identifier.                           *)
    LINES[Y]:=CONCAT(BON,'Slide #: ',W,BOF,'   ','Identification: '
                     ,IDENT);
    IF ANAME THEN PRTAPHID  (* If the option was selected, print *)
    ELSE BEGIN              (* the aphid name record.            *)
```

```
      STR(ANUM,W);      (* Process the aphid name reference number. *)
      LINES[Y]:=CONCAT(LINES[Y],'   Aphid #:',BON,W,BOF,' ',ACOMF);
    END;
    Y:=Y+1;             (* Process the number of aphids on the slide. *)
    LINES[Y]:=CONCAT(MANY,' specimen');
    IF MANY>'1' THEN LINES[Y]:=CONCAT(LINES[Y],'s');
                                        (* Process the aphid morph. *)
    IF MORPH<>'' THEN LINES[Y]:=CONCAT(LINES[Y],'     Morphs:',MORPH);
    IF CNAME THEN    (* If the option was selected, print the *)
                     (* collection data record.               *)
      IF (CNUM<CCONST+1) OR (CNUM>CCONST+LASTCOL) THEN BEGIN
        Y:=Y+1;
        STR(CNUM,W);
        LINES[Y]:=CONCAT(BON,'Coll.#:',W,BOF,'  Not in file.');
      END
      ELSE PRTCOLL
    ELSE BEGIN
      STR(CNUM,W);
      LINES[Y]:=CONCAT(LINES[Y],'     Coll. #:',BON,W,BOF);
    END;
    IF SCOM<>'' THEN BEGIN  (* Process the slide comments. *)
      Y:=Y+1;
      LINES[Y]:=CONCAT('Comments: ',SCOM);
    END;
    Y:=Y+1;          (* Add a blank line. *)
    LINES[Y]:=' ';
  END;
END;  (* PRTSLIDE *)


PROCEDURE SETPRINTER;
(*************************************************)
(*                                               *)
(* INITIALIZE THE PRINTER AND FORMATTING COMMANDS  *)
(*                                               *)
(*************************************************)
BEGIN
  WRITE(PRINTER,CHR(27),'E');  (*SELECT ELITE CHARACTERS*)
  WRITE(PRINTER,CHR(27),'L015');  (*SET LEFT MARGIN AT 1.5 INCH*)
  WRITE(PRINTER,CHR(27),'A'); (*LINE FEED PITCH*)
  UON:='$1B$58';   (*START UNDERLINING*)
  UOF:='$1B$59';   (*STOP UNDERLINING*)
  BON:='$1B$21';   (*BOLD FACE ON*)
  BOF:='$1B$22';   (*BOLD FACE OFF*)
  WRITELN(PRINTER);
END;  (* SETPRINTER *)
```

```
PROCEDURE SORTHOST(REC:INTEGER);
(*********************************************)
(*                                           *)
(* PRINT THE HOST-PLANT NAMES ALPHABETICALLY *)
(*                                           *)
(*********************************************)
BEGIN
  SEEK(PLANT,REC);
  GET(PLANT);
  IF PLANT^.PLNOD<>0 THEN SORTHOST(PLANT^.PLNOD);
  IF PLANT^.PNUM<>REC THEN BEGIN
    SEEK(PLANT,REC);
    GET(PLANT);
  END;
  Y:=0;
  PRTPLANT;
  IF PLANT^.PSYN<>0 THEN BEGIN
    Y:=Y+1;
    LINES[Y]:=CONCAT(BON,'     Synonyms:',BOF);
  END;
  SENDPRT;
  IF PLANT^.PSYN<>0 THEN BEGIN  (* Print the synonym(s) of the aphid *)
    REPEAT                      (* name, if any.                      *)
      SEEK(PLANT,PLANT^.PSYN);
      GET(PLANT);
    UNTIL PLANT^.PSYN=0;
    WHILE PLANT^.PSIM DO BEGIN
      Y:=0;
      PRTPLANT;
      SENDPRT;
      SEEK(PLANT,PLANT^.PBNOD);
      GET(PLANT);
    END;
  END;
  IF PLANT^.PHNOD<>0 THEN SORTHOST(PLANT^.PHNOD);
END;  (* SORTHOST *)


PROCEDURE SORTAPHID(REC:INTEGER);
(*************************************)
(*                                   *)
(* PRINT THE APHID NAMES ALPHABETICALLY *)
(*                                   *)
(*************************************)
BEGIN
  SEEK(APHID,REC);
  GET(APHID);
  IF APHID^.ALNOD<>0 THEN SORTAPHID(APHID^.ALNOD);
  IF APHID^.ANUM<>REC THEN BEGIN
    SEEK(APHID,REC);
    GET(APHID);
  END;
```

```
    Y:=0;
    PRTAPHID;
    IF APHID^.ASYN<>0 THEN BEGIN   (* Print the synonym(s) of the aphid *)
      Y:=Y+1;                      (* name, if any.                     *)
      LINES[Y]:=CONCAT(BON,'     Synonyms:',BOF);
    END;
    SENDPRT;
    IF APHID^.ASYN<>0 THEN BEGIN
      REPEAT
        SEEK(APHID,APHID^.ASYN);
        GET(APHID);
      UNTIL APHID^.ASYN=0;
      WHILE APHID^.ASIM DO BEGIN
      Y:=0;
      PRTAPHID;
      SENDPRT;
      SEEK(APHID,APHID^.ABNOD);
      GET(APHID);
    END;
  END;
  IF APHID^.AHNOD<>0 THEN SORTAPHID(APHID^.AHNOD);
END;  (* SORTAPHID *)


BEGIN   (*HARDCOPY*)
  WRITE(CHR(12));
  GOTOXY(33,0);    (* Display the title *)
  WRITE(CHR(15),'PRINTING A FILE',CHR(14));
  GETCRTINFO;
  RESET(FNAMES,'COLLECT:FILEINFO');
  SETCHAIN('COLLECT:TITLE');
  REPEAT
    REWRITE(PRINTER,'PRINTER:');
  UNTIL IOCHECK('PRINTER');
  ALPHA:=FALSE;
  LSYN:=FALSE;
  SELECT;
  SETPRINTER;
  GOTOXY(0,22);CRT(ERASEOS);
  WRITE(CHR(15),'POSITION THE PAPER AND PRESS SPACEBAR.',CHR(14));
  WW:=GETCHAR([' ']);
  PAGE:=0;
  CASE SEL OF
    'A','a':FOR I:=FIRST TO LAST DO BEGIN  (* Print the collection *)
              SEEK(COLL,I);                (* data file.          *)
              GET(COLL);
              IF PNAME THEN BEGIN
                SEEK(PLANT,COLL^.PNUM);
                GET(PLANT);
              END;
              Y:=0;
              PRTCOLL;
```

```
          SENDPRT;
        END;
'B','b':FOR I:=FIRST TO LAST DO BEGIN  (* Print the slide      *)
          SEEK(SLIDE,I);               (* data file.           *)
          GET(SLIDE);
          IF ANAME THEN BEGIN
            SEEK(APHID,SLIDE^.ANUM);
            GET(APHID);
          END;
          IF CNAME THEN BEGIN
            IF (SLIDE^.CNUM>CCONST) OR (SLIDE^.CNUM<=CCONST+LASTCOL)
              THEN BEGIN
                SEEK(COLL,SLIDE^.CNUM);
                GET(COLL);
            END;
          END;
          IF PNAME THEN BEGIN
            SEEK(PLANT,COLL^.PNUM);
            GET(PLANT);
          END;
          Y:=0;
          PRTSLIDE;
          SENDPRT;
        END;
             (* Print the host-plant name file. *)
'C','c':IF LSYN THEN FOR I:=FIRST TO LAST DO BEGIN
          SEEK(PLANT,I);                 (* Print the list of host- *)
          GET(PLANT);                    (* plant synonyms.         *)
          IF PLANT^.PSIM=TRUE THEN BEGIN
            Y:=0;
            PRTPLANT;
            SENDPRT;
          END;
        END
        ELSE IF ALPHA THEN BEGIN
          SEEK(PLANT,0);
          GET(PLANT);
          SORTHOST(PLANT^.PBNOD);
        END
        ELSE FOR I:=FIRST TO LAST DO BEGIN
          SEEK(PLANT,I);
          GET(PLANT);
          Y:=0;
          PRTPLANT;
          SENDPRT;
        END;
                          (* Print the aphid name file. *)
'D','d':IF LSYN THEN FOR I:=FIRST TO LAST DO BEGIN
          SEEK(APHID,I);     (* Print the list of aphid *)
          GET(APHID);        (* synonyms.               *)
          IF APHID^.ASIM=TRUE THEN BEGIN
            Y:=0;
```

```
            PRTAPHID;
            SENDPRT;
          END;
        END
        ELSE IF ALPHA THEN BEGIN
          SEEK(APHID,0);
          GET(APHID);
          SORTAPHID(APHID^.ABNOD);
        END
        ELSE FOR I:=FIRST TO LAST DO BEGIN
          SEEK(APHID,I);
          GET(APHID);
          Y:=0;
          PRTAPHID;
          SENDPRT;
        END;
  END;
  BYEBYE
END.  (* HARDCOPY *)
```

```
PROGRAM PRTLABELS;
(*******************)
(*                 *)
(* PRINT THE LABELS *)
(*                 *)
(*******************)
USES CHAINSTUFF;
TYPE SETOFCHAR=SET OF CHAR;
    CRTCOMMAND= (ERASEOS,ERASEOL,UP,DOWN,RIGHT,LEFT,LEADIN);
    LONG=INTEGER[6];
VAR COLL:FILE OF RECORD
           CNUM:INTEGER;
           DATE:STRING[12];
           CITY:   STRING[15];
           STATE:  STRING[15];    (* See Table 4 for a description *)
           COUNTRY: STRING[15];   (* of the variables.             *)
           COLLECT: STRING[20];
           PNUM:INTEGER;
           PCOMF:STRING[1];
           LOCAT:STRING[50];
           COLONY:STRING[50];
           COLOUR:STRING[226];
           CCOM:STRING[226];
          END;
  PLANT:FILE OF PACKED RECORD
          PNUM:INTEGER;
          PFAM:STRING[20];
          PGENUS:STRING[20];    (* See Table 3 for a description *)
          PSPEC:STRING[20];     (* of the variables.             *)
          PVAR:STRING[20];
          PDESC:STRING[20];
          PSIM:BOOLEAN;
          PSYN:INTEGER;
          PHNOD:INTEGER;
          PLNOD:INTEGER;
          PBNOD:INTEGER;
         END;
   SLIDE:FILE OF RECORD
           SNUM:INTEGER[6];
           CNUM:INTEGER;
           ANUM:INTEGER;
           ACOMF:  STRING[1];
           MORPH: STRING[12];   (* See Table 2 for a description *)
           IDENT: STRING[20];   (* of the variables.             *)
           MANY:STRING[2];
           SCOM:STRING[65];
          END;
  APHID:FILE OF PACKED RECORD
          ANUM:INTEGER;
          AGENUS:STRING[20];
          ASBGN:STRING[20];
          ASPEC:STRING[20];     (* See Table 2 for a description *)
```

```
        ASBSP:STRING[20];     (* of the variables.            *)
        ADESC:STRING[20];
        ASIM:BOOLEAN;
        ASYN:INTEGER;
        AHNOD:INTEGER;
        ALNOD:INTEGER;
        ABNOD:INTEGER;
      END;
   FNAMES:FILE OF RECORD
          CFILE,PFILE,SFILE,AFILE:STRING;
        END;
   (* See the program 'HARDCOPY' for a description of the variables. *)
   LINES:ARRAY[1..10] OF STRING;
   Y,LASTCOL,CCONST,I,FIRST,LAST,IORES:INTEGER;
   WW:CHAR;
   BON,BOF,UON,UOF:STRING[6];
   PRINTER:TEXT;
   CRTINFO: PACKED ARRAY[CRTCOMMAND] OF CHAR;
   PREFIXED:ARRAY[CRTCOMMAND] OF BOOLEAN;
(*$I-*)


PROCEDURE GETCRTINFO;
(*****************************************)
(*                                       *)
(* READ THE SCREEN CONTROL CHARACTERS FROM *)
(* THE SYSTEM.MISCINFO FILE              *)
(*                                       *)
(*****************************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE CRT(C:CRTCOMMAND);
(*******************)
(*                 *)
(* SCREEN COMMANDS *)
(*                 *)
(*******************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE BYEBYE;
(*************************************)
(*                                   *)
(* UPDATE FILEINFO AND CLOSE THE FILES *)
(*                                   *)
(*************************************)

(* See the program 'HARDCOPY' for the listing and comments. *)
```

```
PROCEDURE  TRLONG(VAR QQ:STRING;VAR TEMP:LONG);
(*****************************************************)
(*                                                   *)
(* CONVERT THE VALUE OF A STRING INTO A LONG INTEGER *)
(*                                                   *)
(*****************************************************)

(* See the program 'SLIDEAPHID' for the listing and comments. *)


 FUNCTION GETCHAR(OK:SETOFCHAR):CHAR;
 (*************************************)
 (*                                   *)
 (* GET A CHARACTER FROM THE KEYBOARD *)
 (*                                   *)
 (*************************************)

 (* See the program 'TITLE' for the listing and comments. *)


FUNCTION IOCHECK(F:STRING):BOOLEAN;
(*************************************)
(*                                   *)
(* VERIFY THE STATUS OF THE LAST I/O *)
(* OPERATION AND PRINT AN ERROR      *)
(* MESSAGE IF THE OPERATION WAS NOT  *)
(* SUCCESSFUL                        *)
(*                                   *)
(*************************************)

(* See the program 'HOST' for the listing and comments. *)


PROCEDURE SELECT;
(************************************************)
(*                                              *)
(* SELECTION OF THE SLIDE NUMBERS TO BE PRINTED *)
(*                                              *)
(************************************************)
VAR LASTPL,LASTAP,LASTSL:INTEGER;
    SCONST:INTEGER[6];


PROCEDURE OPEN (A:CHAR;L:INTEGER);
(*********************************)
(*                               *)
(*     NAME AND OPEN FILES       *)
(*                               *)
(*********************************)

(* See procedure 'SELECT' in the program 'HARDCOPY' for *)
```

```
(* the listing and comments.                                    *)


PROCEDURE INITLAB;
(*************************************************)
(*                                               *)
(* READ THE SLIDE REFERENCE NUMBERS TO BE PRINTED *)
(*                                               *)
(*************************************************)
VAR L1,L2:INTEGER[6];
    OK:BOOLEAN;
BEGIN
  GOTOXY(15,6);CRT(ERASEOS);
  WRITE('First desired slide number:');
  GOTOXY(15,8);
  WRITE('Last desired slide number:');
  REPEAT
    REPEAT
      GOTOXY(43,6);CRT(ERASEOL);
      READLN(LINES[1]);
      TRLONG(LINES[1],L1);
      OK:=FALSE;
      IF L1>SCONST THEN
        IF (L1-SCONST)<32767 THEN
          IF (TRUNC(L1-SCONST))<=LASTSL THEN OK:=TRUE;
    UNTIL OK=TRUE;
    REPEAT
      GOTOXY(42,8);CRT(ERASEOL);
      READLN(LINES[1]);
      TRLONG(LINES[1],L2);
      OK:=FALSE;
      IF L2>SCONST THEN
        IF (L2-SCONST)<32767 THEN
          IF (TRUNC(L2-SCONST))<=LASTSL THEN OK:=TRUE;
    UNTIL OK=TRUE;
  UNTIL L1<=L2;
  FIRST:=TRUNC(L1-SCONST);
  LAST:=TRUNC(L2-SCONST);
END;  (* INITLAB *)


BEGIN    (* SELECT *)
  OPEN('B',4);
  INITLAB;
  OPEN('A',10);
  OPEN('C',12);
  OPEN('D',14);
END;  (* SELECT *)
```

```
PROCEDURE SENDPRT;
(*************************************************)
(*                                               *)
(* SEND THE FORMATTED INFORMATION TO THE PRINTER *)
(*                                               *)
(*************************************************)

(* See the program 'HARDCOPY' for the listing and comments. *)


PROCEDURE SETPRINTER;
(*******************************************************)
(*                                                     *)
(* INITIALIZE THE PRINTER AND THE FORMATTING COMMANDS  *)
(*                                                     *)
(*******************************************************)
BEGIN
  WRITE(PRINTER,CHR(27),'E');  (*SELECT ELITE CHARACTERS*)
  WRITE(PRINTER,CHR(27),'L003');(*SET LEFT MARGIN *)
  WRITE(PRINTER,CHR(27),'A'); (*LINE FEED PITCH*)
  UON:='$1B$58';   (*START UNDERLINING*)
  UOF:='$1B$59';   (*STOP UNDERLINING*)
  BON:='$1B$21';   (*BOLD FACE ON*)
  BOF:='$1B$22';   (*BOLD FACE OFF*)
  WRITELN(PRINTER);
END;  (* SETPRINTER *)


PROCEDURE RIGHTLAB;
(*******************************)
(*                             *)
(* FORMAT THE RIGHT SIDE LABEL *)
(*                             *)
(*******************************)
(* The right side label contains:                          *)
(*      the slide reference number printed in bold face on  *)
(*          the first line,                                 *)
(*      the aphid name; the correct name will be used if the *)
(*                      slide data refers to a synonym and  *)
(*                      the aphid name will be underlined.  *)
(*                      The program will use as few lines as *)
(*                      possible to print the information.  *)
(*      the determinator's name on the 9th line.            *)
VAR L,J:INTEGER;
    W:STRING;
```

```
PROCEDURE PART_I;
(**********************)
(*                    *)
(* RIGHTLAB, PART ONE *)
(*                    *)
(**********************)
BEGIN
  Y:=Y+1;
  STR(SLIDE^.SNUM,W);  (* Process the slide reference number. *)
  LINES[Y]:=CONCAT(BON,'Slide#:',W,BOF);
  Y:=Y+1;              (* Process the aphid genus. *)
  LINES[Y]:=CONCAT(UON,APHID^.AGENUS,UOF);
  L:=LENGTH(APHID^.AGENUS);
  IF APHID^.ASBGN<>'' THEN  (* Process the aphid subgenus. *)
    IF L+3+LENGTH(APHID^.ASBGN)<=20 THEN BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' (',UON,APHID^.ASBGN,UOF,')');
      L:=L+3+LENGTH(APHID^.ASBGN);
    END
    ELSE BEGIN
      Y:=Y+1;
      LINES[Y]:=CONCAT('(',UON,APHID^.ASBGN,UOF,')');
      L:=2+LENGTH(APHID^.ASBGN);
    END;
  IF APHID^.ASPEC<>'' THEN W:=APHID^.ASPEC
  ELSE W:='sp.';       (* Process the aphid species. *)
  IF L+1+LENGTH(W)<=20 THEN BEGIN
    LINES[Y]:=CONCAT(LINES[Y],' ',UON,W,UOF);
    L:=L+1+LENGTH(W);
  END
  ELSE BEGIN
    Y:=Y+1;
    LINES[Y]:=CONCAT(UON,W,UOF);
    L:=LENGTH(W);
  END;
  IF APHID^.ASBSP<>'' THEN  (* Process the aphid subspecies. *)
    IF (L+7+LENGTH(APHID^.ASBSP))<=20 THEN BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' subsp.',UON,APHID^.ASBSP,UOF);
      L:=L+7+LENGTH(APHID^.ASBSP);
    END
    ELSE BEGIN
      Y:=Y+1;
      LINES[Y]:='subsp.';
      IF LENGTH(APHID^.ASBSP)<15 THEN
        LINES[Y]:=CONCAT(LINES[Y],UON,APHID^.ASBSP,UOF)
      ELSE BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT(UON,APHID^.ASBSP,UOF);
      END;
      L:=LENGTH(LINES[Y])-12;
    END;
  END;  (* PART_I *)
```

```
  BEGIN  (* RIGHTLAB *)
  PART_I;
  IF SLIDE^.ACOMF<>'' THEN
    IF (L+2)<=20 THEN BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' ',SLIDE^.ACOMF);
      L:=L+2;
    END
    ELSE BEGIN
      Y:=Y+1;
      LINES[Y]:=SLIDE^.ACOMF;
      L:=1;
    END;              (* Process the aphid describer. *)
  IF (L+1+LENGTH(APHID^.ADESC))<=20 THEN
    LINES[Y]:=CONCAT(LINES[Y],' ',APHID^.ADESC)
  ELSE BEGIN
    Y:=Y+1;
    LINES[Y]:=APHID^.ADESC;
  END;               (* Process the determinator's name. *)
  IF Y<8 THEN FOR J:=Y+1 TO 8 DO LINES[J]:='';
  IF LENGTH(SLIDE^.IDENT)>16 THEN BEGIN
    IF Y<8 THEN LINES[8]:=CONCAT(BON,'Det:',BOF);
    LINES[9]:=SLIDE^.IDENT;
  END
  ELSE LINES[9]:=CONCAT(BON,'Det:',BOF,SLIDE^.IDENT);
  Y:=9;
END;  (* RIGHTLAB *)


PROCEDURE LEFTLAB;
(******************************)
(*                            *)
(* FORMAT THE LEFT SIDE LABEL *)
(*                            *)
(******************************)
(* The left side label contains:                                  *)
(*     the collection reference number; printed on the first line *)
(*          in bold face.                                         *)
(*     the country, province and city; printed on as few          *)
(*          lines as possible.                                    *)
(*     the date; printed alone on one line.                       *)
(*     the collector's name; alone on one line.                   *)
(*     the host-plant genus, species and variety.                 *)
(*          will be underlined and printed on as few lines as     *)
(*          possible.                                             *)
VAR L,I:INTEGER;
    W:STRING;
```

```
PROCEDURE PART_A;
(*********************)
(*                   *)
(* LEFTLAB, PART ONE *)
(*                   *)
(*********************)
BEGIN
  Y:=Y+1;
  W:='';
  STR(SLIDE^.CNUM,W);  (* Process the collection reference number. *)
  LINES[Y]:=CONCAT(BON,'Coll.#:',W,BOF);
  Y:=Y+1;              (* Process the country. *)
  LINES[Y]:=CONCAT(BON,'Loc:',BOF,COLL^.COUNTRY);
  L:=LENGTH(COLL^.COUNTRY)+4;
  IF COLL^.STATE<>'' THEN   (* Process the province. *)
    IF (L+1+LENGTH(COLL^.STATE))<=20 THEN BEGIN
      LINES[Y]:=CONCAT(LINES[Y],' ',COLL^.STATE);
      L:=L+1+LENGTH(COLL^.STATE);
    END
    ELSE BEGIN
      Y:=Y+1;
      W:=COLL^.STATE;
      FOR I:=1 TO 4 DO W:=CONCAT(' ',W);
      LINES[Y]:=W;
      L:=LENGTH(W);
    END;
  IF COLL^.CITY<>'' THEN   (* Process the city. *)
    IF (L+1+LENGTH(COLL^.CITY))<=20 THEN
      LINES[Y]:=CONCAT(LINES[Y],' ',COLL^.CITY)
    ELSE BEGIN
      Y:=Y+1;
      W:=COLL^.CITY;
      FOR I:=1 TO 4 DO W:=CONCAT(' ',W);
      LINES[Y]:=W;
    END;
  Y:=Y+1;          (* Process the date. *)
  LINES[Y]:=CONCAT(BON,'Date:',BOF,COLL^.DATE);
  Y:=Y+1;
  IF LENGTH(COLL^.COLLECT)<16 THEN
    LINES[Y]:=CONCAT(BON,'Coll:',BOF,COLL^.COLLECT)
  ELSE IF LENGTH(COLL^.COLLECT)<19 THEN
         LINES[Y]:=CONCAT(BON,'C:',BOF,COLL^.COLLECT)
       ELSE LINES[Y]:=COLL^.COLLECT;
END;  (* PART_A *)


BEGIN  (* LEFTLAB *)
     (* Continue if the collection reference number is within range. *)
  IF (SLIDE^.CNUM>CCONST) AND (SLIDE^.CNUM<=CCONST+LASTCOL) THEN BEGIN
    PART_A;
    Y:=Y+1;    (* Process the host-plant genus. *)
    IF PLANT^.PGENUS<>'' THEN BEGIN
```

```
      IF LENGTH(PLANT^.PGENUS)<16 THEN BEGIN
        LINES[Y]:=CONCAT(BON,'Host:',BOF,UON,PLANT^.PGENUS,UOF);
        L:=5+LENGTH(PLANT^.PGENUS);
      END
      ELSE IF LENGTH(PLANT^.PGENUS)<19 THEN BEGIN
             LINES[Y]:=CONCAT(BON,'H:',BOF,UON,PLANT^.PGENUS,UOF);
             L:=2+LENGTH(PLANT^.PGENUS);
           END
           ELSE BEGIN
             LINES[Y]:=CONCAT(UON,PLANT^.PGENUS,UOF);
             L:=20;
           END;           (* Process the host-plant species. *)
      IF PLANT^.PSPEC<>'' THEN W:=PLANT^.PSPEC
      ELSE W:='sp.';
      IF L+1+LENGTH(W)<=20 THEN BEGIN
        LINES[Y]:=CONCAT(LINES[Y],' ',UON,W,UOF);
        L:=L+1+LENGTH(W);
      END
      ELSE BEGIN
        Y:=Y+1;
        LINES[Y]:=CONCAT(UON,W,UOF);
        FOR I:=1 TO 5 DO BEGIN
          IF LENGTH(W)<20 THEN LINES[Y]:=CONCAT(' ',LINES[Y]);
          W:=CONCAT(' ',W);
        END;
      END;
      IF PLANT^.PVAR<>'' THEN   (* Process the host-plant variety. *)
        IF(L+5+LENGTH(PLANT^.PVAR))<=20 THEN BEGIN
          LINES[Y]:=CONCAT(LINES[Y],' var.',UON,PLANT^.PVAR,UOF);
          L:=5+LENGTH(PLANT^.PVAR);
        END
        ELSE BEGIN
          Y:=Y+1;
          IF LENGTH(PLANT^.PVAR)<17 THEN
            W:=CONCAT('var.',UON,PLANT^.PVAR,UOF)
          ELSE W:=CONCAT(UON,PLANT^.PVAR,UOF);
          FOR I:=1 TO 5 DO
            IF LENGTH(W)<32 THEN W:=CONCAT(' ',W);
          LINES[Y]:=W;
          L:=LENGTH(LINES[Y])-12;
        END;
      IF COLL^.PCOMF<>'' THEN  (* Process the collector's name. *)
        IF L<19 THEN LINES[Y]:=CONCAT(LINES[Y],' ',COLL^.PCOMF);
    END;
  END
  ELSE BEGIN  (* If the collection reference number is out of range. *)
    Y:=Y+2;
    LINES[Y-1]:=('COLLECTION NUMBER');
    LINES[Y]:=('OUT OF RANGE');
  END;
END;  (* LEFTLAB *)
```

```
PROCEDURE BACKFEED;
(*****************************************)
(*                                       *)
(* MOVE BACK TO THE FIRST LINE OF THE LABEL *)
(*                                       *)
(*****************************************)
VAR K:INTEGER;
BEGIN
  WRITE(PRINTER,CHR(27),'r');
  FOR K:=1 TO Y DO WRITELN(PRINTER);
  WRITE(PRINTER,CHR(27),'f');
END;  (* BACKFEED *)


PROCEDURE LABELS;
(********************)
(*                  *)
(* PRINT THE LABELS *)
(*                  *)
(********************)
VAR I,J,N:INTEGER;
    ONE:BOOLEAN;
BEGIN
  ONE :=FALSE;
  N:=1;
  FOR I:=1 TO 10 DO LINES[I]:='';
  FOR I:= FIRST TO LAST DO BEGIN
    SEEK(SLIDE,I);      (* Read the slide data record. *)
    GET(SLIDE);
    (* Check the range and read the collection data record. *)
    IF (SLIDE^.CNUM>CCONST) AND (SLIDE^.CNUM<=CCONST+LASTCOL) THEN BEGIN
      SEEK(COLL,SLIDE^.CNUM);
      GET(COLL);
    END;
    SEEK(APHID,SLIDE^.ANUM);  (* Read the aphid name record. *)
    GET(APHID);
    IF APHID^.ASIM=TRUE THEN  (* Find the correct name if a synonym. *)
      REPEAT
        SEEK(APHID,APHID^.ABNOD);
        GET(APHID);
      UNTIL APHID^.ASIM=FALSE;
    SEEK(PLANT,COLL^.PNUM);   (* Read the host-plant name record. *)
    GET(PLANT);
    IF PLANT^.PSIM=TRUE THEN  (* Find the correct name if a synonym. *)
      REPEAT
        SEEK(PLANT,PLANT^.PBNOD);
        GET(PLANT);
      UNTIL PLANT^.PSIM=FALSE;
    ONE:=NOT (ONE);
    Y:=0;
    LEFTLAB;
```

```
    IF ONE THEN WRITE(PRINTER,CHR(27),'L003') (* Set the left margin. *)
    ELSE WRITE(PRINTER,CHR(27),'L049');
    SENDPRT;
    BACKFEED;
    Y:=0;
    RIGHTLAB;
    IF ONE THEN WRITE(PRINTER,CHR(27),'L026') (* Set the left margin. *)
    ELSE WRITE(PRINTER,CHR(27),'L072');
    SENDPRT;
    IF ONE THEN BACKFEED
    ELSE BEGIN
      N:=N+1;              (* Change page after 5 lines of two pairs of *)
      IF N>5 THEN BEGIN    (* labels were printed.                      *)
        N:=1;
        WRITE(PRINTER,CHR(12));
      END                (* If the page is not filled, skip three lines*)
      ELSE FOR J:=1 TO (3+(9-Y)) DO WRITELN(PRINTER);
    END;
  END;
END;  (* LABELS *)


BEGIN   (*PRTLABELS*)
  WRITE(CHR(12));
  GOTOXY(33,0);
  WRITE(CHR(15),'PRINTING LABELS',CHR(14));
  GETCRTINFO;
  RESET(FNAMES,'COLLECT:FILEINFO');
  REPEAT
    REWRITE(PRINTER,'PRINTER:');
  UNTIL IOCHECK('PRINTER');
  SETCHAIN('COLLECT:TITLE');
  SELECT;
  SETPRINTER;
  GOTOXY(0,22);CRT(ERASEOS);
  WRITE(CHR(15),'POSITION THE PAPER AND PRESS SPACEBAR.',CHR(14));
  WW:=GETCHAR([' ']);
  LABELS;
  BYEBYE
END.  (* PRTLABELS *)
```